A Study of Mobile Phone Ad Hoc Networks via Bluetooth with different Routing Protocols

Author: Martha Kamkuemah

Supervisor: Dr. Hanh Le

Dissertation presented to the

Department of Computer Science
Faculty of Science
University of Cape Town

in partial fulfilment of the requirements for the degree of

Master of Science

May 30, 2013

Plagiarism Declaration

I know the meaning of plagiarism and declare that all of the work in the dissertation, save for which is properly acknowledged, is my own work.

Signature

Abstract

The growth of mobile computing is changing the way people communicate. Mobile devices, especially mobile phones, have become cheaper and more powerful, and are able to run more applications and provide networking services. Mobile phones use fixed cellular infrastructure such as base stations and transmission towers to enable users to share multimedia content and access the internet at any time or place. However, using the internet is costly. Therefore, one of the solutions is to create impromptu ad hoc networks to share information among users. Such networks are infrastructureless and self-organising, much like mobile ad hoc networks.

This dissertation therefore investigates how mobile phones with low-power Bluetooth technology can be used to create ad hoc networks that connect mobile phones and allow them to share information. The mobile phones should be able organise themselves for multi-hop communication. Routing becomes important in order to achieve efficiency in data communication. Several existing routing protocols were developed and evaluated for this network to determine how efficiently they deliver data and deal with network disruptions such as a device moving out of transmission range.

Representative routing protocols in mobile ad hoc networking, peer-to-peer networks and publish/subscribe systems were evaluated according to performance metrics defined in the research, namely total traffic, data traffic, control traffic, delay, convergence time, and positive response. Prototypes for Nokia phones were developed and tested in a small ad hoc network. For practical networking setup, a simple routing protocol that uses the limited mobile phone resources efficiently would be better than a sophisticated routing protocol that keeps routing information about the network participants.

"We can make our plans,
but the Lord determines our steps"
– Proverbs 16:9 (NLT)

To my family and supervisor
– thank you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Mobile phone penetration in Africa rose from 5% in 2008 to over 30% in 2012. By 2008, over 50% of South African adults had a mobile phone [47]. According to the Allied Business Intelligence research group, Bluetooth is the most widely implemented wireless connectivity technology, with up to 906 million Bluetooth-enabled mobile phones shipped worldwide in 2010 [1].

Bluetooth is a short-range, low-power technology originally designed to replace wire cables between mobile devices. Bluetooth allows nearby mobile devices to exchange data without relying on telecommunication infrastructure. Networking with Bluetooth is used in many applications. Bluetooth-based headsets for voice communication have became quite popular [33]. Multimedia file sharing in a peer-to-peer (P2P) manner [44] is another popular use of Bluetooth technology. Bluetooth is also used in pervasive tracking to approximate the locality of other Bluetooth-enabled devices [12]. Networking with Bluetooth is used in mobile health systems to transfer patient information to doctors in hospitals [15, 42, 81]; special announcements can be broadcast to Bluetooth-enabled mobile phones in convention centres, conferences, and classrooms; Bluetooth ad hoc networks are also used in rapid deployment of electromagnetic identification readers [41].

1.1 Motivation

Many people in South Africa use their mobile phones for voice communication, text messaging and internet access. While Bluetooth is still used for file sharing between devices in close proximity, data exchange using Bluetooth is not as prevalent as using cellular infrastructure to send text or instant messages. However using these cellular services introduces extra cost. This research considers the creation of a mobile ad hoc network, henceforth referred to as a Bluetooth ad hoc network, consisting of mobile phones connected in a ubiquitous manner. While traditional cellular networks rely on fixed infrastructure, ad hoc networks are formed without central administration and fixed infrastructure. The devices rely on Bluetooth wireless channels to share information. Such a network allows the mobile devices to be self-organising and self-configuring, connecting other devices out of transmission range. Devices should be able to look for and exchange information through the network without necessarily knowing which devices have the required information. Devices should be able to forward data in a multi-hop manner to devices out of transmission range by using intermediary devices. Each device is therefore responsible for maintaining routes to other devices and making routing decisions. Routing should use the limited resources of the mobile devices efficiently

while at the same time adapting to the unpredictable topology changes caused by mobile devices moving out of transmission range or being switched off during transmission [23].

Academic research has explored the use of Bluetooth ad hoc networking: for example, with vehicular ad hoc network systems that intelligently disseminate traffic information via Bluetooth. Bluetooth is used to broadcast traffic information between moving vehicles and control centres [76]. Bluetooth ad hoc networks use the P2P concept that was initially designed for overlay networks over the internet. Mobile ad hoc networks share the same characteristics. This research will explore how P2P capabilities can be implemented in the Bluetooth ad hoc networks for file sharing.

1.2 Research Questions

The research focuses on building a Bluetooth ad hoc network consisting of mobile phones. The research investigates the feasibility of creating a multi-hop ad hoc network and study how routing performs in such a network using Bluetooth protocol features. The research therefore investigates the following research questions:

1. *What is the feasibility of peer-to-peer file sharing in a Bluetooth ad hoc network?*

Peer-to-peer networking requires that devices have similar capabilities, such as hardware, and implement data lookup techniques on each device. These characteristics are shared by mobile ad hoc networks, except that mobile devices in mobile ad hoc networks have limited battery power, computation and memory. To maximise resource usage, Bluetooth, which uses little power to connect devices, is considered for file sharing. However, Bluetooth communication imposes a master/slave relationship that allows one device to be a master and control communication with other devices, called slaves devices.

A master device and slave devices form a piconet. A master device can be a slave in another piconet, forming a bridge between piconets. Interconnected piconets form multi-hop ad hoc networks called scatternets [81]. This research will investigate how this Bluetooth feature can be used to form a multi-hop Bluetooth ad hoc network that performs data lookup or file sharing in a P2P manner.

2. *How do existing routing protocols perform in the multi-hop Bluetooth ad hoc network?*

Routing is the process of transferring data from a source node to a destination node were each participating node is responsible for making a routing decision about where to forward data. The Bluetooth ad hoc network requires a way to route data through this network. Because the network uses concepts from the P2P and mobile ad hoc networking paradigm, the research considers how routing protocols implemented in these networks will perform in the Bluetooth ad hoc network.

Broadcasting, a popular routing technique in P2P overlay networks, simply forwards data to devices in a network if the data has never been encountered. Devices using Broadcasting maintain only routing information of devices one hop away.

Because the Bluetooth ad hoc network described is concerned with file sharing, the research also investigates how content-based [34] routing (also known as semantic-routing), used mostly in publish/subscribe systems, will perform in such a network. Content-based routing ignores destination addresses and routes packets based on content. Content-based routing could possibly be suitable for this network because it provides more flexibility by loosely connecting communication nodes. Devices do not have to be aware of who sent and received the data. This suggests that content-based routing could also be suitable for routing in the Bluetooth ad hoc network where disconnections can happen at any time when mobile phones are switched off or move out of the transmission range.

Routing is a well researched and challenging problem in mobile ad hoc networks. Efficient routing is challenging because node movement breaks links and causes network partitions and data loss. To address the goal of finding the routing protocol suitable for this network, the routing protocol should incur low overhead during communication between nodes, and efficiently use of the limited resources of mobile phones, such as battery power and storage. To achieve low overhead, minimise message delivery delay and maximize throughput, mobile ad hoc network (MANET) reactive routing protocols such as Advanced On-demand Distance Vector Routing (AODV) and Dynamic Source Routing (DSR) only initiate route discovery when they need to send a data packet. This research question therefore investigates how existing routing protocols perform in the Bluetooth ad hoc network.

1.3 Methodology

In order to answer the research questions, a constructive prototyping methodology was employed: *a*) prototype design; *b*) system development; *c*) quantitative experiments; and *d*) evaluation.

The prototype was designed and developed on the Java 2 Platform Micro Edition (J2ME). The J2ME platform was used because it is supported on many low-end mobile devices. The first prototype design implemented the two on-demand routing protocols - AODV and DSR - based on specifications by the Internet Engineering Task Force (IETF)[1]. This prototype used to study the feasibility of P2P file sharing and routing between mobile phones using Bluetooth. The prototype was tested on a test bed of five Nokia N96 mobile phones. The data collected showed much variability due to technical challenges during testing.

[1] `www.ieft.org`

The second prototype design implemented additional routing protocols, namely content-based routing and broadcasting, and was evaluated in a controlled environment. Experimentation took the form of emulation. Emulation was chosen because of the difficulty encountered in monitoring the test bed, and the lack of simulation mobility models that closely resemble the characteristics of the Bluetooth ad hoc network described.

Performance metrics were defined and routing performance evaluated accordingly. Evaluation proceeded with the interpretation of results to determine how well the routing protocols performed in the Bluetooth ad hoc network. The interpretation of results helped determine which routing protocol was more suitable for the real world.

Evaluation revealed that broadcasting and content-based routing outperformed AODV and DSR in terms of delay and convergence times metrics chosen. AODV and DSR had the highest total traffic, attributed mainly to control traffic. These protocols also had the lowest delay and convergence times because of the routing decisions such as updating route cache tables and determining the next forwarding address. On the other hand, broadcasting and content-based routing had the least positive response because these protocols made little attempt at trying to receive responses from destinations to which requests had been sent. These observations suggest that broadcasting and content-based routing seem better suited for information dissemination in sparse, stable mobile phone Bluetooth ad hoc networks.

1.4 Outline of Dissertation

The rest of the dissertation is organised as described below. Chapter 2 reviews relevant background and related work. It presents the evolution of networking architectures from the wired client/server architecture, mobile ad hoc networks, the P2P networking paradigm, and finally P2P over mobile ad hoc networks. The chapter reviews the characteristics and routing techniques employed in some of these networks. Chapter 3 presents the design of a mobile application deployed on mobile phones that allows them to form an ad hoc network and route data using the Bluetooth technology. The design of the mobile application and routing protocols is presented in a top-down manner in this chapter. The implementation of the mobile application and the routing protocols are also presented using a top-down approach. Chapter 4 presents the emulation environment set up to obtain data and analysis of these data. The chapter presents and discusses the results obtained for the routing protocols and analyses which routing protocol will be suitable and efficient for use in a mobile phone Bluetooth ad hoc network. Chapter 5 concludes the dissertation by summarising the research, answering the research questions, and making a recommendation for further study.

Chapter 2

Background

This chapter discusses networking architectures from wired to wireless networks. The chapter begins by discussing a common application in networking architectures - that of file sharing. The chapter then briefly reviews client/server architectures that popularised file sharing. With the evolution of wireless communication technologies such as WiFi and Bluetooth, wireless mobile ad hoc networks are now being used to disseminate information by connecting devices out of transmission range. These networks are becoming more popular because they do not require pre-existing infrastructure to connect devices. This chapter also reviews the routing protocols used in these networks to deliver information efficiently. Information between nodes in a mobile ad hoc network is shared in a P2P manner. This chapter also reviews P2P networks, their topologies and lookup techniques. The chapter also looks at earlier attempts to implement P2P lookup techniques in mobile ad hoc networks. Because this is a feasibility study, content-based networks are also reviewed. The chapter concludes with a discussion of Bluetooth technology, its capabilities and limitations.

2.1 File Sharing

File sharing is probably the most popular application of mobile ad hoc networking. It is a way of providing access to digitally stored information such as multimedia (music, video, audio and image files), documents and electronic books. Common methods to provide access to this information is through centralised servers, distributed P2P and mobile ad hoc networks. File sharing systems mostly work on wired networks. File sharing for these networks require the construction of search algorithms for transmitting queries and search results as the development of matching techniques to match user queries. File sharing systems such as Gnutella, Napster, KaZaa and BitTorrent are developed for the wired internet [60].

Rajagopalan and Shen[60] discuss popular P2P file sharing systems. Napster uses client/server architecture. A server stores the information files while clients register with the server in order to download information. Systems such as Gnutella and KaZaa allow users to share files directly. They blur the distinction between clients, servers and routers as each peer individually fulfils these roles.

2.2 Client/Server Architectures

Client/server architectures organise network resources, such as computers, into hierarchical structures. Servers administer tasks to clients. Servers are made up of hardware components that run specialised software. Clients request services from the servers. Popular systems that use the client/server architecture are file servers, database servers, transaction servers, web servers and domain name servers.

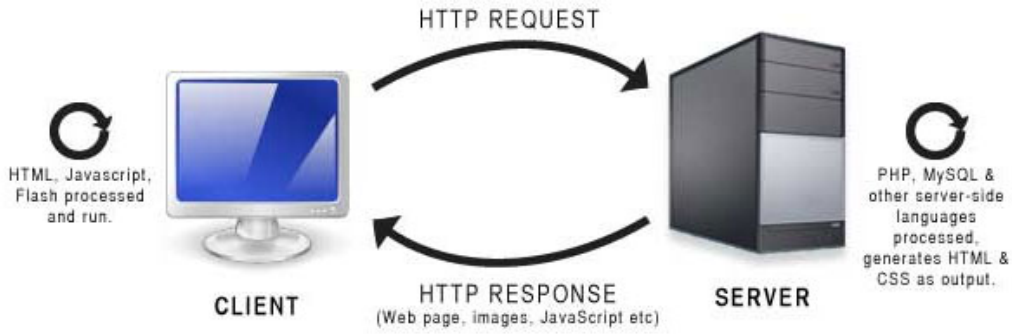


Figure 2.1: Web server architecture [11]

Figure 2.1 illustrates a two-tier architecture. Client/server architectures often evolve from two-tier architectures as shown in Figure 2.2. Figure 2.2 shows an example of a multi-tier system that has evolved over time. These systems employ many features at different levels to deal with user requests.

Separation of client and server functions using a design concept called modularity increases the flexibility and usability of network resources. Usability is improved with user-friendly client interfaces, which allow users to issue requests to servers.

Communication in client/server networks uses the unique IP addresses assigned to participants. The IP addresses are authenticated by the server; after that authorised exchange of message formats defined by the IETF[1] can take place. Two types of transport protocols are used to transfer data from one participant to another, namely connection-oriented and connectionless protocols. Connection-oriented protocols guarantee that a message will arrive at its intended destination and communication will terminate afterwards. Connectionless protocols do not guarantee that a message sent by the client will be received by the server, hence communication does not terminate. By far the most popular connectionless protocol is the Internet Protocol. This protocol connects large numbers of local and wide area networks that comprise the internet.

[1] http://www.ietf.org/

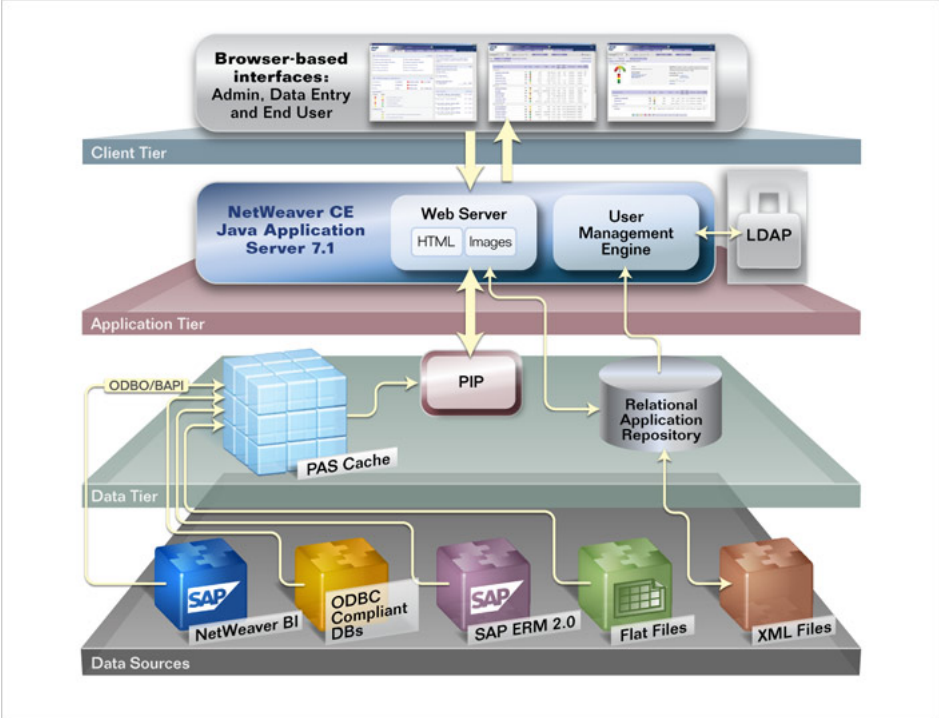


Figure 2.2: Multi-level tier architecture [11]

2.2.1 Summary

Client/server architectures often have a single point of failure. When a server crashes, client requests cannot be processed. Servers can only process a certain number of client requests, after which adding more requests begins to slow down the performance of the architecture. Client-server architectures therefore improve scalability using distributed computing. However the wireless environment presents challenges to the implementation of a client-server architecture. Mobile devices are usually physically small and have low processing power and memory capacity compared to client-server hardware. For the purposes of this project, it is infeasible to implement a client-server architecture in a mobile ad hoc network in view of the static nature of servers. Servers need to be available constantly, waiting for client requests. Servers also rely on fixed infrastructure and are not highly mobile.

2.3 Mobile Ad Hoc Networks

MANETs are a collection of self-organising wireless mobile devices. The mobile devices leave and join the network in an ad hoc manner; the nodes organise themselves by communicating with nearby nodes. To communicate with nodes far away, nodes use limited transmission range wireless to create multi-hop links. In this way nodes use intermediate nodes as routers to relay information to destination nodes out of transmission range. By relaying information, MANETs incorporate routing

capabilities that enable efficient information sharing over inherently unreliable wireless connections [2, 4, 15, 19, 72]. Despite the inherent unreliability of the wireless connections, MANETs provide interesting opportunities to create rapidly deployable applications for filesharing, data mules and delay tolerant transient networks to upload or download information [40].

MANETs were widely used mainly on battlefields and in emergency search and rescue operations, e-health systems and sensor networks [19, 29].

Wireless links between nodes use technologies such as Bluetooth or WiFi. Wireless links are inherently unreliable owing to node mobility. Nodes join and leave the network in an ad hoc fashion. Because of the limited transmission range of the wireless standards used, nodes in MANETs can serve as routers to relay packets on behalf of other nodes, thus causing nodes to communicate in a multi-hop manner. Because of high node mobility and inherently unreliable wireless links, network partitioning and merging are common, resulting in highly dynamic topologies [7]. The mobile devices have limited battery power, processing power and memory. Intelligent routing algorithms that use these limited resources efficiently and are capable of adapting to the dynamic topologies of these networks are highly regarded.

Inherently unreliable wireless connections and highly dynamic topology pose challenges to information sharing in MANETs. Therefore, applications developed for these networks should be able to handle the dynamicity of MANETs. In the following sections, technical challenges and possible solutions related to routing, node and resource discovery are reviewed.

One of the key issues in MANETs is how to route packets efficiently. Routing algorithms must be energy efficient, have low delivery latency, an excellent packet delivery rate, and must be adaptable and scalable. Routing must use the limited resources of MANETs efficiently and adapt to the changes in the dynamic network topologies. Because of the inherently unreliable nature of wireless links, traditional wired-network link-state and distance vector routing algorithms are not suitable for mobile networks [72, 64]. Various routing protocols have been designed to cope with the different requirements. Some routing algorithms are reactive; they update routing information while the network is alive and use route maintenance mechanisms to fix transmission errors caused by disconnected or broken links. Proactive routing algorithms such as destination-sequenced distance-vector (DSDV) and optimised link state routing (OLSR) determine the routes to all destinations at start-up and periodically update this information. Nodes have a full view of the network at all times. However periodic updates generate heavy overhead traffic and it is for this reason that reactive or on-demand routing algorithms such as AODV and DSR are used, which determine a route only when it is needed by flooding the network with route request messages.

Generally, table-driven protocols reduce control overhead traffic (i.e. lower message latency) at the

cost of latency when finding the route to the destination node. It is expected that the control messages generated may not reach their destinations in a network where links are broken and network partitioning occurs.

Table-driven routing protocols (e.g. DSDV and OLSR) allow each node to maintain one or more tables containing routing information to every other node in the network. Each node updates its routing tables by periodically exchanging control messages with other nodes in order to maintain a consistent and up-to-date view of the network [56]. In source-driven routing protocols (e.g. DSR and AODV) nodes do not maintain tables with routing information. Nodes only initiate a route discovery mechanism when a source node needs to send traffic to a destination node.

Routing in mobile ad hoc networks is quite broad. This section looks at unicast routing associated with the routing techniques to be discussed in the following sections.

2.3.1 Proactive Routing Protocols

The table-driven DSDV algorithm maintains several routing tables at each node. The routing tables provide information about every possible destination in the network. The routing tables are used to find a single path to a destination node using the distance vector shortest path algorithm [4]. In order to find the shortest path, routing tables also contain sequence numbers for every destination [64]. Therefore, routing tables are updated with the latest information. This guarantees that routes are fresh. Routing tables are updated either by using a full route update approach, where all tables of a node exchange information with other nodes, or in an incremental fashion containing information of the last full update [4]. The two update approaches try to reduce overhead messages, which use up bandwidth. However overhead messages increase with the periodic updates required. This protocol does not scale well for large networks [4].

With OLSR, each node maintains network topology information by periodically exchanging link-state messages with other nodes [4]. OLSR uses a multipoint replaying strategy to minimise the size of control messages and number of rebroadcast nodes during route updates. Selected sets of nodes called multipoint relays are selected during route updates to rebroadcast packets. To select multipoint relays, each node periodically broadcasts a list of its one-hop neighbours using hello messages. From the list of one-hop neighbours, each node selects a subset of one-hop neighbours [4]. “Each node determines an optimal route (in terms of hops) to every known destination using its topology information (from the topology table and neighbouring table), and stores this information in a routing table. Therefore, routes to every destination are immediately available when data transmission begins” [4].

2.3.2 Reactive Routing Protocols

In DSR, each node maintains a routing cache. When a source node tries to send a packet, it first checks its route cache for a route to the destination. If a valid route to the destination is found in the cache, it is used. Otherwise, the source node initiates the route discovery mechanism by broadcasting a route request to its neighbours. The route request packet contains the full address of the source node, the destination address and a unique sequence number for loop detection [64]. The neighbour node receiving the request checks its own routing cache for a route to the destination. If the receiving node is the intended destination, it copies the routing information in the route request packet into a reply packet, and sends the reply packet to the source. If the receiving node is not the intended destination, it checks its route cache for a route to the destination node, then adds its address to the route request packet and then forwards the request to its neighbours. A major advantage of the DSR protocol is that nodes first check their route caches before they initiate the route discovery mechanism. However this protocol does not scale well for large networks because of the amount of overhead carried in the route request packet. Therefore in a large, highly dynamic network, the amount of overhead consumes bandwidth.

AODV, which is similar to DSR, differs from DSR in that it periodically checks if routing paths are still available. It does so by maintaining routing tables for nodes located only on active paths . Nodes that do not lie on active paths do not participate in updating routing tables or maintaining routing information. AODV uses two mechanisms to maintain routing tables, route discovery and route maintenance. Route discovery is initiated whenever a node has no established route to its neighbours. Route discovery is initiated by broadcasting a route request packet. Every node receiving the route request packet copies the source address into its route cache and forwards the message to its neighbours [38]. AODV is scalable in large dynamic networks, but extra delays and more bandwidth consumption could be introduced as the network grows [4, 58, 72].

The two main functions performed by AODV and DSR, namely route discovery and route maintenance, are illustrated in Figures 2.3 and 2.4.

2.3.3 Broadcasting

During broadcasting, or simply flooding, a node sends a packet to all forward neighbouring nodes in the network. When an intermediary node receives a packet, it checks its routing table to determine if it has seen the packet before. If it has, it simply ignores the packet or else forwards the packet to all its forward neighbours. Broadcasting is often used in MANETs for route discovery and route maintenance[75].

Williams and Camp [75] discuss the different types of broadcasting in MANETs: blind flooding, probability-based, area-based, and neighbour knowledge approaches.

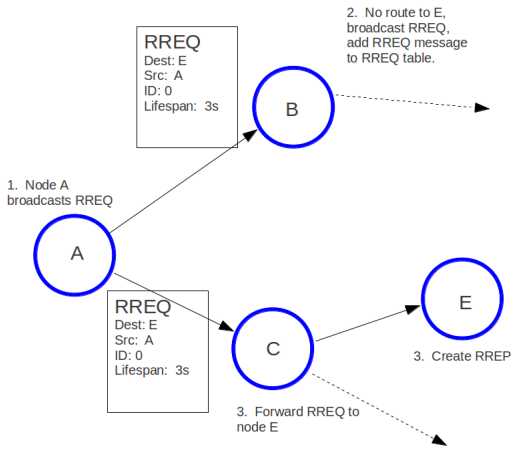


Figure 2.3: AODV and DSR route discovery begins by discovering all routes within one-hop distance. This process is made easy by Bluetooth's service discovery functionality. Although nodes are not connected, a node is able to sense which devices are within its communication range.

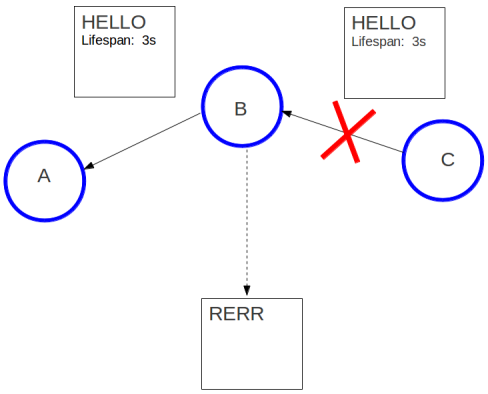


Figure 2.4: AODV route maintenance sends out periodic HELLO messages. If a node does not receive a HELLO message after a specified time, a route error message is created. A simple route maintenance mechanism is designed as illustrated in figure 2.4. When the connection to node C is broken, node B triggers a route error message.

Each node rebroadcasting the same message creates message redundancy. As the network size increases, so does the redundancy. Redundancy causes high bandwidth consumption and increases packet collisions, which can lead to more transmissions. As more collisions occur, contention for the wireless medium increases. Redundancy, collision and contention are all part of the broadcast storm problem [37]. It is therefore important for this project that broadcast algorithms reduce the number of transmissions in the network.

Simple techniques such as expanding ring searching are used to reduce performance degradation caused by the broadcast storm problem.

2.3.4 Mobility

The inherently dynamic nature of mobile ad hoc networks necessitates having dependable methods to determine the reliability of wireless links. Reliability of wireless links is negatively influenced by mobility, which causes communication links to break, resulting in network fragmentation. In proactive routing protocols, route discovery fixes these problems without the transmission nodes knowing about the link break. Reactive routing protocols only discover link failure once they try to transmit data. Reactive routing protocols thus experience a huge delay when transmitting data and this affects the quality of service of the network.

Mobility models are differentiated according to spatial and temporal dependencies. Spatial dependency measures how two nodes are dependent on each other. If two nodes are moving in the same direction, they have a high spatial dependency. Temporal dependency measures how current velocity is dependent on previous velocity. Within these dependencies are four commonly used mobility models, namely *Random Waypoint*, *Random Point Group Mobility*, *Freeway Mobility* and *Manhattan Mobility Model*. When the Random Waypoint model is used, a node chooses a destination and moves towards it with randomly chosen uniform velocity. Random Waypoint is a popular mobility model [8] but does not really take into consideration the spatial and temporal dependencies required of mobility models. Freeway Mobility emulates the behaviour of nodes on a freeway. Nodes are restricted to lanes in a freeway and current velocity depends on previous velocity. With Random Group Mobility, nodes are divided into groups. A group leader determines the group's motion behaviour [68]. Random Group Mobility has high spatial dependency between nodes because the speed and direction for each node is derived by a slight deviation from that of the group leader. High spatial dependency leads to fewer link breakages and a less dynamic topology. The Manhattan Mobility Model, on the other hand, exhibits high spatial and temporal dependency. The model emulates node movement on a street map. Mobility models discussed by Son et. al [68] show that current velocity depends on previous velocity and the direction is chosen probabilistically.

These models use various mobility metrics to represent mobility. Tsumochi et. al [73] divide mobility metrics into three groups, *node*, *link* and *neighbours* metrics. The node group focuses on a node, its velocity for example, while the link looks at connectivity between two nodes. The neighbours metric focuses on mobility information between neighbouring nodes. Xu et. al [77] developed a mobility metric, link availability, which refers to the probability that a currently available link will be active at a particular time in the future. This metric was explored in previous work by [50] and [79]. Xu et. al's link availability model [77] is a statistical model that epoch lengths, or the length interval during which a node moves at a constant speed in a constant direction, to predict link availability between two mobile nodes. On the other hand, Haas [28] predicts link availability

between two nodes located at (x_1, y_1) and (x_2, y_2) and separated by a distance D. Nodes move uniformly in the area covered by distance D according to their defined mobility model. The existence of a link is modelled as a stochastic random variable with exponential probability density function:

$$P(D) = \int_0^\infty \frac{1}{\lambda} e^{-\lambda t} \, \mathrm{d}t. \tag{2.1}$$

Often simulation programs are used to model link reliability. It is important however to understand how mobility is represented in order to get an accurate depiction of the behaviour of the network. Xu *et al.*[77] use a stochastic process to determine the mobility metric for a particular link and network average metric. The authors use Markov Chain Modelling to determine the link availability. Jiang *et al.* [32], on the other hand, used a probability exponential distribution function to determine link availability. Jiang *et al.* [32] indicate that link availability prediction can be used to simulate node movement in simulation programs such as ns, Glomosim, etc.

2.3.5 Summary

In summary, MANETs are:

- *Dynamic* networks in which nodes move randomly. Nodes can also be static, active or inactive. Thus the network topology changes arbitrarily at random times. Therefore, constant connectivity is not guaranteed. The other distinguishing feature of the topology is how the network is formed: Bluetooth nodes only hear each other when they form a master-slave pair. This is in contrast to wireless local area networks (WLANs) where any two nodes in proximity can hear each other's transmissions.
- *Multi-hop.* Routes between nodes have multiple intermediary nodes.
- *Bandwidth-constrained.* The network's wireless links have low bandwidth capacity.
- *Power-constrained.* The nodes have limited battery power. To conserve energy, the application developed must efficiently consume battery power.
- *Limited security.* MANETs rely on security built into the wireless network protocol to reduce security threats.

Reactive and proactive routing algorithms require that destination nodes be known before transmission of a message begins [56]. Destinations known beforehand are somewhat too specified, as the proposed network focuses more on content. However the routing protocols provide insight into the types of routing tables to be constructed to reduce message overhead. These algorithms also illustrate the need to maintain routing information about nodes in close proximity.

2.4 Pure Peer-to-Peer Networks

Unlike client/server systems, pure P2P networks consist of heterogeneous participants that function as both servers and clients, with no dedicated server in the system. The participants distributed are over pre-existing underlying network infrastructure such as the internet and have no central control [4, 25, 38, 39, 63, 64, 71]. The participants form a virtual overlay topology that hides the underlying physical network [30]. The participants cooperate and share their resources such as processing power, disk storage and bandwidth to fulfil a task such as file sharing [57]. One of the first P2P networks was the internet, in which an internet host could telnet/ftp any other internet host [3]. However P2P systems were popularised by file-sharing applications such as Napster, Gnutella and Freenet [3].

P2P networks are categorised into two types depending on the organisation of participants, namely structured and unstructured P2P networks. Structured P2P networks form well-defined overlay topologies, which make searching more efficient [43]. Unstructured P2P networks form random overlay networks, which raise challenges for searching and routing algorithms.

2.4.1 Structured Peer-to-peer Networks

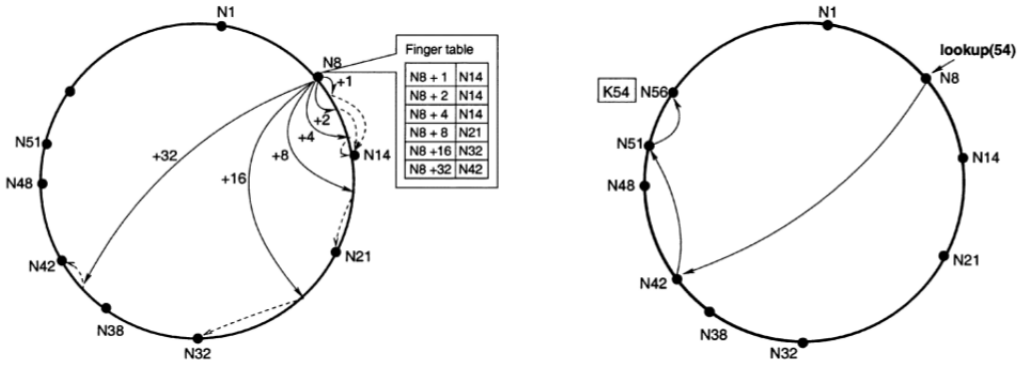
Kellerer et. al [36] present the most common type of structured P2P network: the distributed hash table (DHT). DHT creates a keyspace by hashing values (either file names or contents) into keys and storing the keys and their associated values as (key, value) pairs. Participating nodes are assigned a randomly selected identifer or ID from the keyspace. There are several well-known P2P networks. Chord and Content Addressable Network are reviewed. Chord, a distributed lookup protocol, uses a circular keyspace. Each node is responsible for a subset of (key, value) pairs in the keyspace. A lookup for a key returns the node storing the object with that key or the node whose node ID equals or follows key or K in the identifier space. The lookup or search services are said to be efficient, with (key, value) pairs returned with probabilistic certainty. Kellerer et. al [36] show that most DHT-based lookup protocols resolve lookups with logarithmic messages.

Chord

Chord uses a circular keyspace called ring of size area modulo 2^m. Chord then uses consistent hashing to arrange m-bit keys and node IDs in a circular space called a Chord ring [71]. m is sufficiently large to make the probability of keys hashing to the same identifier negligible. This also allows nodes to join and leave the network with minimal interruption [45, 67].

Nodes join the Chord network by requesting a successor from a bootstrap node. Once connected in the network, the new node notifies its successor and predecessor nodes to update their successor and predecessor pointers, and inherently their finger tables. The process of updating successors,

predecessors and finger tables is called stabilisation and is performed periodically to ensure correctness in the Chord network.



(a) Finger table (b) Key look up

Figure 2.5: Finger table lookup [71]

Chord performs lookup for internet-based applications. To perform lookup services, Chord maintains routing information in routing tables called finger tables. Each node maintains m entries - keys or node IDs - in its finger table. The ith entry in the finger table is the first node s that succeeds node n by at least 2^{i-1} in the keyspace. Node s is called the successor peer of key K, labelled *successor(K)*, i.e. s = successor(n + 2^{i-1}) where $1 \leq$ i $\leq$ m. The example in Figure 2.4.1 shows the finger table of node 8. Suppose node 8 wants to find a successor to key 54. K54 is stored at node N56. The lookup algorithm speeds up its operations by going to the largest finger of node 8 that precedes 54, which is node 42; node 8 will then ask node 42; to resolve the query. In turn, node 42 will determine the largest finger in its finger table that precedes 54, i.e., node 51. Finally, node 51 will discover that its own successor, node 56, succeeds key 54, and thus, will return node 56 to node 8. Chord protocol periodically runs stabilisation protocol to update successor pointers and finger tables affected when nodes join and leave the network [45]. Each node maintains a finger table of size $O(logn)$, and searching path length $O(logn]$, where n equals the total number of nodes and files [21, 70].

The Chord lookup operation is scalable. Consistent hashing allows peers to join and leave the network with minimal interruption. This tends to balance the load on the system, since each peer receives roughly the same number of keys.

Content Adressable Network (CAN)

CAN uses a multi-dimensional Cartesian coordinate keyspace to create a virtual overlay network. The Cartesian keyspace is dynamically divided between peers. A key is deterministically mapped onto a point P in the coordinate space using a uniform hash function [45]. The lookup protocol

retrieves an entry corresponding to key K by using the same hash function to map K onto the point P. A new peer joining the network looks up a bootstrap peer IP address in the DNS of the CAN domain. The bootstrap peer provides addresses of randomly chosen peers in the network. The new peer randomly selects a zone to join and sends a join request directed to that zone. Peers in the network route the request to the right peer in that zone. Once the node managing that zone receives the request, it splits its zone in half and allocates the other half to the new peer [45]. If the requesting peer and its neighbours do not own the point P, the request is routed through the CAN network until it reaches the peer where zone P lies.

Each CAN peer maintains a routing table containing the IP address and coordinates of neighbouring nodes. A greedy routing algorithm is used to route messages towards a destination using those coordinates that are closer to the destination. The lookup cost is $O(d \times N^{\frac{1}{d}})$ with routing table size $O(N)$ [21, 45, 70]. Like Chord, CAN uses an additional maintenance protocol to remap the identifier space onto nodes periodically. Ratnasamy et al. [61] improve CAN performance by using "weighted round-trip-times" from a node to each of its neighbours. So, in order to forward a message to a given destination, the message is sent along the path with a low round trip time, which favours lower latency paths, and in turn helps application level CAN routing avoid unnecessarily long hops.

2.4.1.1 Conclusion

A common feature in structured P2P networks is the use of hashed key values to determine the location of data in the organised topology or keyspace. This allows structured P2P systems to perform simple, fast lookup functions for finding data. Often the key mismatch in the overlay network and the underlying physical network is a cost that cannot be sustained. Structured P2P networks require that the network be stable to maintain the topology. Network stability is not a required characteristic of the envisioned network.

DHT systems are fault-tolerant and scalable owing to the hash function that evenly distributes IDs and keys between participating nodes, but DHT-based P2P applications still need to deal with the fact that they completely ignore the topology of the underlying physical network, possibly turning a single hop in the overlay network into many physical hops in the underlying physical network. Because of logical ID routing, two overlay neighbours in a DHT-based overlay network may not be connected in the underlying physical network. This topology mismatch could cause serious search (latency) delays, which could affect lookup services because the route may no longer be available.

2.4.2 Unstructured Peer-to-Peer Networks

This section reviews fully distributed unstructured P2P networks that have no centralised control nor any defined shape of the overlay topology. Gnutella and its disadvantages [62] and Freenet [17, 18] are such examples. The network is formed as nodes join and leave arbitrarily without any prior knowledge of the overlay and underlying network topologies. Content search in these

networks usually involves query flooding, where a search query is propagated to all neighbours within a certain range. These unstructured networks are robust in allowing nodes to join and leave the network without seriously affecting the operation of the network. Flooding data requests consumes high bandwidth. The search mechanisms are not efficient and scalable [46]. However these networks have low overhead in overlay maintenance, which makes them more suitable for transient networks and dynamic content [46].

Gnutella

Gnutella is a decentralised file searching P2P system [3, 14] which uses the Gnutella Protocol to search for files. Nodes use bootstrapping to join a Gnutella network. Gnutella provides host caches, which contain IP addresses to existing nodes. The new node connects to an existing node in the host cache by opening a transmission control protocol connection and engaging in a handshake procedure. Once connected to the network, a node communicates with neighbouring nodes by broadcasting PING (requests information about another node) and query messages. Broadcasting duplicates query messages at each node and these are further relayed to multiple nodes. Duplicate messages therefore consume excessive network resources such as bandwidth and make the network less scalable. To reduce the consumption of network resources, nodes cache PONG (response) messages and supply them as replies to PING messages. Gnutella also controls time-to-live (TTL) or the maximum number of hops of a message to minimise the extent of duplication in the network. Another improvement to Gnutella uses ultrapeers. Berkes [11] implements the concept of ultrapeers by dividing nodes into super nodes and regular nodes. Super nodes act as proxies for connecting nodes. They are well connected and take up most of the burden of routing network messages for regular nodes. Regular nodes connect to the network using super nodes as entry points.

The flood-based searching technique allows nodes to maintain information about neighbouring nodes. However flooding data queries is bandwidth intensive. One way to control the flood of query messages is to set a TTL counter for each message broadcast. The TTL is set to a small value initially and the query message is forwarded until the TTL value reaches zero. This is called the expanding ring search. If a query gets no response, another query message with a longer TTL is broadcast again.

Freenet

Freenet is another decentralised P2P system for the "publication, replication and retrieval of data files". It is designed to protect the privacy of the creators and users of data, and protect data by encrypting file contents [3].

Clarke [17, 16, 18] introduced the concept of sharing hard-drive space between devices and accessing it using content hash keys. Nodes participating in the network share a portion of their hard drive (called the datastore) with other nodes. Data stored in the network are associated with a key that

is used to determine where the data are to be stored. The key is also used to authenticate the data when these are transferred. Freenet uses two types of keys: a content hash key is generated from the data and a signed subspace key used to publish signed documents in the private subspace of a publisher with an asymmetric key-pair. Each node therefore also has a routing table consisting of a set of (key, pointer) pairs that can be used to search for documents. The pointer points to a node with a copy of the file [18].

Clarke [17, 16, 18] describes how nodes join the network for the first time by using Freenet opennet connections. Opennet connects nodes to publicly known seednodes already connected to the network. Later, nodes switch to Darknet connections after building trust relationships with other nodes. Trust relationships are built with those nodes participating in successful search requests, for instance. Routing uses a purely greedy algorithm: at each step of the routing procedure the desirability of the neighbours is ordered according to the proximity of their IDs to the route key. Search requests are thus forwarded from node to node, with each node making a local routing decision where to send the query next until the query is satisfied or a TTL value expires (TTL again refers to the number of hops a request is permitted to travel before it is discarded). This means that requests are routed through the network rather than broadcast. Routing is also semantic-based, that is, users need to know the exact name of the file they are searching for in order to retrieve the data. When data are found, the node will replicate the data into its own datastore and create an entry in its routing table associating the actual data source with the requested key. Replicating data at multiple nodes can improve data availability and response time, but replicating data in a network consisting of limited storage capacity mobile nodes can degrade the performance of the nodes.

Freenet, according to its inventor, Clarke [17, 16, 18], responds adaptively to usage patterns. A node tends to receive requests for keys similar to keys it holds in its datastore. It therefore tends to specialise in clustering files with similar keys, and becomes better at answering queries from its routing tables. This effect improves data availability and search response time in the network.

The routing algorithm maintains a data store at each node containing files recently used while handling requests and inserts, and a routing table that contains keys for data objects and nodes that might be able to handle the request. A Freenet request consists of a desired key and a hops-to-live (HTL) value, the hop count before the request expires [6]. When a node receives a request, it checks its data store for the key entry. If the data are found, the node returns the data and a reference to itself to the previous node, which adds the reference to its routing table and passes the data up the chain. If the data are not found, the closest key to the key being sought is used. The request is passed to the node with the closest key and the HTL value is decremented. The HTL value continues to be decremented until it expires and a DataNotFound [6] message is returned to the requester. Freenet maintains the requester's anonymity by passing data back in a hop-by-hop

fashion instead of passing data directly to the requester. If the HTL is high, data retrieval can be quite slow. The longer the path, the more unreliable it becomes. Freenet exhibits small-world properties: specialisation [6]. A node tends to specialise in certain key spaces, that is, if it is queried for similar keys more often, it builds a data store of those keys and dissimilar keys expire.

2.4.2.1 Conclusion

Broadcasting or flooding is common in the above-mentioned unstructured P2P networks. Broadcasting however does not require maintaining topology information, which is the nature of mobile ad hoc networks, as nodes come and go dynamically. The envisioned network reduces broadcast costs by minimising message overhead.

The flood-based searching technique used in Gnutella is not always efficient. Floods that start with a small TTL value can be used. The TTL, much like the HTL in Freenet, determines how far and how long a searh query lives in the network. If a search is unsuccessful, a new search request is initiated again by the source of the message with a larger TTL, which propagates further into the network. This is called an expanding ring search.

Expanding ring searches intend to reduce flood overhead in multi-hop networks and is more efficient in terms of bandwidth usage when a query file is found with the smallest TTL. However if the TTL value keeps increasing, this approach introduces extra overhead [46].

To improve bandwidth usage further, random walks can be used that randomly select a neighbour node to which to forward a query message. This approach cuts down on message overhead when the queried resource is popular. The random walk fails when the queried resource is scarce and held by only a few nodes [46].

2.5 Peer-to-Peer over Mobile Ad Hoc Networks

Researchers ([48]) have sought ways to implement the P2P paradigm in mobile ad hoc networks. One method was simply implementing P2P protocols over MANETs. However studies have shown that simply implementing a P2P system on top of a mobile ad hoc network affects routing performance [48] and increases complexity. Further developments led to the use of multiple cross-layer implementations that map the nodes in the overlay network closely to the nodes in the underlying physical network. This section discusses previous work concerning cross-layer implementations.

2.5.1 P2P Platforms

7DS, a filesharing application level protocol, enables exchange of information between peer devices and the internet. It conducts service and device discovery efficiently. In order to do so, it uses a proxy server with a search engine to handle requests between peer devices. The major drawback

with 7DS is that it uses a centralised server [57, 69].

2.5.2 Broadcast over Broadcast

Mapping virtual overlay to the network layer is the simplest adaptation of P2P over MANETs. This approach facilitates direct interaction between the applications and the underlying wireless networks. This approach can simply be implemented by creating an application that is deployed in a MANET.

Flooding is used to search for and route packets in this network. The advantage of this approach is that it is simple to implement and works best in small networks. However, flooding results in heavy traffic, which consumes bandwidth and the limited mobile resources. Moreover, with routing costs equal to $O(n)$, this approach is simply not scalable in dense networks [21] but permissible in sparse networks.

2.5.3 DHT over Broadcast

Simply overlapping a P2P overlay on top of a MANET can cause inefficient resource discovery. A DHT-based searching protocol may be efficient in a wired network, but once introduced in a mobile environment, it can introduce implementation complexities. For instance, to maintain the correctness of routing tables in Chord over MANET, the stabilisation procedure may need to be triggered frequently owing to high node mobility. Ding et al. [21], estimate that the cost of the complexity introduced to the routing algorithm is $O(nlogn)$. This is due to the fact that routing protocols in MANETs introduce a complexity of $O(n)$. When combined with the efficient searching complexity of $O(logn)$ introduced in Chord, routing performance degrades.

2.5.4 Cross-layer Optimisations

To solve the node mobility challenge and dynamic topology of MANETs, cross-layer optimisations are used, which map the underlying physical network topology of the MANET closely with the application-level topology created by the P2P overlay. This is advantageous because topology changes in the network layer due to the dynamic nature of mobile nodes can be filtered up to the application layer, the network layer can perform searching and routing, it reduces overhead control redundancy incurred in both P2P and MANET networks, and MANET security features can improve the P2P network [65].

Reusing existing P2P and routing protocols to build a cross-layer protocol stack minimises the administrative effort and improves performance. Schollmeier et al. modify DSR to create enhanced DSR, by adding a Gnutella-like data search protocol at the application layer and changing the DSR protocol by adding new request and reply types and providing a way to find peers other than by

IP address [65].

Optimized Routing Independent Overlay Network (ORION), a P2P file sharing protocol designed for MANETs, combines application level querying with network routing to reduce search overhead incurred by mismatched virtual and physical topologies [39]. ORION uses AODV and provides application routing. However application level routing causes unnecessary overhead. To minimise application level routing, and improve searching and routing in ORION, ORION+ [2], an enhanced version of ORION, adds new functionality to the routing mechanisms. Routing tables are modified to store two timestamp values - a time when the route request message was sent out and a time when the route reply was received. So available paths are rearranged according to the roundtrip time. Roundtrip times therefore represent path quality, i.e. the shorter the roundtrip the better the path is, and are used as a basis for ordering availables paths so the best one can be selected.

2.5.5 Summary

Combining P2P technologies with MANETs is advantageous because of their autonomous nature and lack of reliance on a fixed network infrastructure. Their combination also minimises the effort of creating new routing protocols.

2.6 Content-based Networks

The growth of wireless technologies and increasing use of mobile devices allow mobile device users to form spontaneous ad hoc networks to disseminate information to devices in close proximity. Traditionally, destination-based routing protocols such as AODV, DSR, OLSR are used to route data in such networks. The publish and subscribe paradigm is well suited to disseminate information in loosely coupled, dynamic networks such as mobile ad hoc networks. Traditional publish and subscribe systems rely on fixed infrastructure of reliable brokers to deliver events from sources to interested users. Publish and subscribe systems have advantages in the mobile ad hoc environment. Their asynchronous and decoupled nature make them more suitable for use in mobile ad hoc networks and the multicasting nature of publish and subscribes systems makes these systems scalable.

Publish and subscribe systems use group-based, subject-based or content-based addressing. Subject-based systems tag content with a short description, or subject, of its content. The subject can be an arbitrary string. User requests are defined in terms of the subject. The user request is matched to the publisher's events. Content-based systems, on the other hand, allow the subscriber more flexibility and control by allowing the user to express interest in arbitrary queries over the content of events. Instead of relying on the publisher to group events into groups or subjects, the subscriber defines queries called subscriptions.

The content-based publish and subscribe paradigm [9] is flexible and suitable for asynchronous communication in dynamic ad hoc networks. These networks provide decoupled communication and asynchronous exchange of messages between nodes. Decoupling means that nodes in the network do not know each other, and asynchrony shows that nodes do not need be connected at the same time to exchange information. Because of these characteristics, the publish/subscribe paradigm is well suited to communicating with nodes in MANETs that require the exchange of information in an opportunistic manner.

However, the flexibility of content-based systems makes design and implementation in mobile ad hoc networks more challenging. Intuitively, since subscriptions can be complex, the matching process between events and subscriptions is harder than in subject-based systems [31].

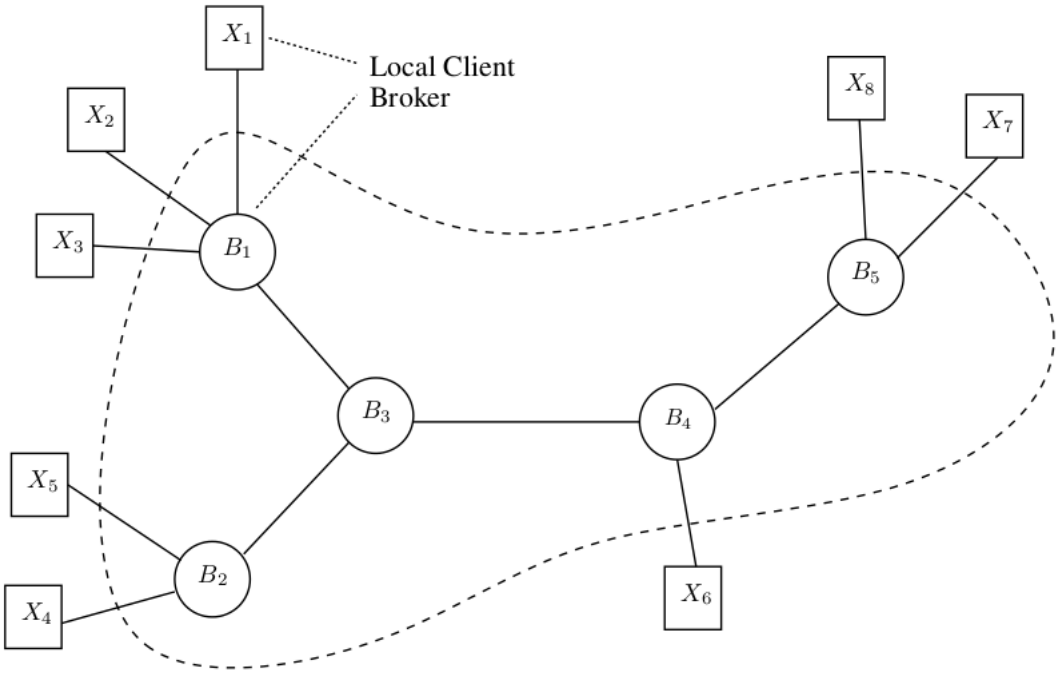


Figure 2.6: Distributed publish/subscribe network [9]

In order for nodes to publish and subscribe to information, a broker topology is established. Figure 2.6 shows how brokers are usually organised. Broker nodes act as routers between client nodes. Client nodes either publish or subscribe to content. Event notification systems such as Gryphon and SIENA [54] use the above topology to form an overlay network on top of existing infrastructure. Multiple brokers form a spanning tree of the network topology by connecting to one another.

Huang et al. [31] discuss various architectures to adapt publish and subscribe systems for the

mobile environment and methods to increase availability and reliability when faced with failures and network partitions. An ad hoc network adaptation places brokers on mobile devices. Instead of using previous knowledge about existing brokers, a broker joining the network discovers those brokers one hop away and connects to them.

Publish/subscribe networks use advertisements, subscriptions and events (also known as *publications and notifications*) as messages to store content. A subscription message expresses a node's interest in a particular event. Muhl [52, 31] describes subscriptions as filters that have the form $F(n) \longrightarrow \{true, false\}$. An event matches a subscription when $F(n)$ evaluates to true. Muhl [53] describe filters as boolean expressions that consist of predicates combined by the boolean operators.

In the meantime, an event message is a set of attributes $\{A_1, ..., A_n\}$ where each A_i is a name, value pair (n_i, v_i). Like the subscription message, an advertisement is also a filter issued by a publisher to indicate its intention to publish future events. The details of the semantics of subscriptions and advertisements are important, especially for achieving efficient routing.

Not only are the semantics for subscriptions important, but so too knowledge of the neighbouring devices. The authors Frey and Roman [22] introduce the concept of context to include aspects such as awareness of one-hop neighbours and link availability. They acknowledge that context changes in response to the dynamic nature of MANETs and inherently unreliable nature of wireless links.

2.6.1 Content-based Routing

Content-based routing algorithms share three entities: a publisher, subscriber and broker. They also share three types of messages: advertisements, events and subscriptions. Content routing algorithms maintain routing tables for these messages. However, it is not a strict requirement for publish and subscribe systems to maintain advertisements. A publisher broadcasts an advertisement indicating the types of events, or *publications*, it will publish in the future. Flooding is usually used in mobile ad hoc networks, as it is the easiest way to disseminate information without the high cost of maintaining topological information. Advertisement messages are stored in an advertisement routing table. The messages also indicate the hops travelled. A subscriber would then send a subscription message expressing interest in a certain event. The advertisement table is used to forward the subscription message to those publishers that can match the subscription. For example, Muhl [53] defines a subscription as a tuple $AF_i = (n_i, Op_i, V_i)$ where n_i is an attribute name, Op_i is a test operator and $V_i = c_{i1}, ..., c_{il}$. If an event does not contain an attribute with n_i than AF_i evaluates to false. Therefore, an event e matches F if it satisfies the values in the tuple.

Nodes on the path between the subscriber to the publisher store the subscriptions in their subscription tables. Finally, events published by the publisher are sent via the reverse path of the subscriptions to interested subscribers. The popular event notification service SIENA uses two

principles to build reverse paths to subscribers, namely subscription forwarding and advertisement forwarding [13]. If a routing algorithm does not use advertisement, routing paths are built by the path subscription messages travel. Advertisement forwarding builds the paths for the subscriptions, which in turn sets the paths for the events. Once advertisements have been propagated through the network, the brokers use the reverse paths to propagate subscriptions.

Routing algorithms use optimisations such as covering, merging and being context-aware to improve routing performance. A covering relation reduces the size of subscription messages. In Muhl's work [52], a filter (or subscription) F_1 covers F_2, i.e. $F_1 \supseteq F_2$ if and only if $N(F_1 \supseteq N(F_2)$. Therefore if $N(F_1) \supseteq N(F_2)$ and $n \notin N(F_1)$, then $n \notin N(F_{2)}$; and if $n \in N(F_2)$ then $n \in N(F_1)$.

Merging, on the other hand, finds the minimum number of subscriptions and advertisements that are representative of the set of filters. In this way, merging reduces the size of routing tables. By Muhl's [53] definition, a filter F covers a set of filters $\{F_1, ..., F_n\}$ if and only if $N(F) \supseteq N(F_1)...N(F_n)$. This relation is said to be a perfect merger if the equality holds, otherwise it is an imperfect merger. If F is a perfect merger, it becomes a covering relation.

2.6.2 Summary

In this section the foundations of content-based routing were discussed. Unlike MANET routing algorithms, content-based routing algorithms are ideally suited to provide the desired communication driven by content rather than explicit addresses by the sender. The sender only learns about nodes one hop away. In this way, the complexity created to maintain knowledge about the whole topology is eliminated. The proposed use of mobile phones in the proposed system makes it suitable to use in applications such as news distribution in disaster areas, service discovery, data sharing and distributed games.

2.7 The Link Layer Protocol: Bluetooth

The Open Systems Interconnection data link layer transfers data between neighbouring network devices. It is subdivided into two layers: the *logical link layer* with multiplexing mechanisms that allow several network protocols to coexist [82] with a multipoint network and be transported over the same network media; and the *media access control layer* which provides channel access control, addressing and multiple access mechanisms to the physical layer. Channel access control, or multiple access protocol, allows several network devices connected to the same physical medium to share it. Shared physical media include bus networks, ring networks and wireless networks. The most widespread use of multiple access protocols is contention-based Carrier sense multiple access with collision detection (CSMA/CD). Mobile phone standards use CSMA as the underlying channel access method. The addressing mechanism uses physical device addresses or MAC addresses to

identify a network device.

This section discusses the Bluetooth protocol, its use in data transmission in ad hoc networks, and the advantages and disadvantages of using Bluetooth communication for this research project. Bluetooth [49] is a short-range, low-cost wireless technology that uses radio technology. Bluetooth capability is embedded on numerous types of devices (PDAs, mobile phones, PCs), data peripherals (mice, keyboards, joysticks, cameras, digital pens, printers, local area network access points), audio peripherals (headsets, speakers, stereo receivers), and embedded applications (automobile power locks, grocery store updates, industrial systems, musical instrument digital interface) [49]. Bluetooth allows devices to form ad hoc networks called wireless personal networks called *piconets*. A piconet is a cluster of up to eight devices. One device is the designated master device, the others slaves. Interconnected piconets form a much larger network called a *scatternet*. Scatternets form the backbone of a MANET and enable devices out of communication range to exchange data via multiple devices.

To form a scatternet, a slave device in one piconet is used as a bridge node in another piconet. Figure 2.7 [2] shows two piconets connected by a bridge device, slave 4/master B.

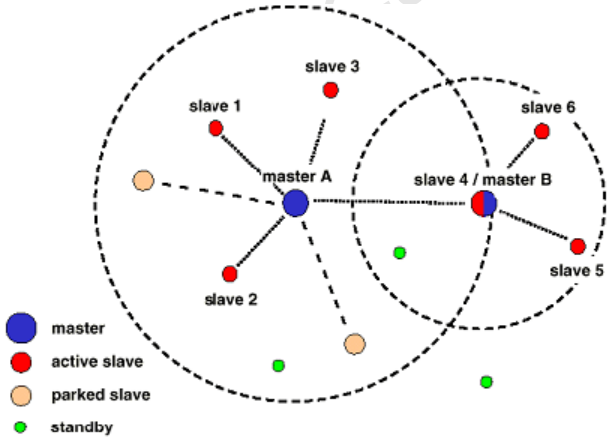


Figure 2.7: Piconet [49]

[2] http://www.developer.nokia.com/Community/Wiki/File:Piconet.png

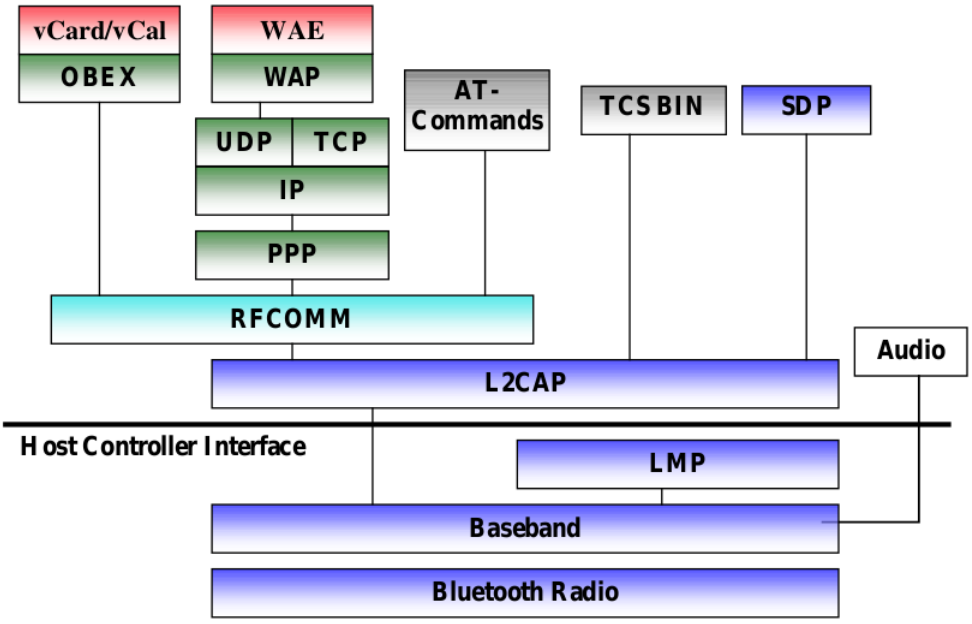


Figure 2.8: Bluetooth protocol stack [49]

Figure 2.8 shows the various protocols that make up the Bluetooth protocol stack. The stack comprises core Bluetooth protocols such as Link Manager Protocol (LMP) and Logical Link Control and Adaptation Protocol (L2CAP), and adopted protocols such as OBEX (Object Exchange Protocol) and user datagram protocol (UDP). LMP does remote device discovery, connects devices and performs authentication. L2CAP provides connection-oriented and connectionless data services to upper layer protocols with protocol multiplexing capability, segmentation and reassembly operation. L2CAP permits higher level protocols and applications to transmit and receive L2CAP data packets up to 64 kilobytes in length.

Bluetooth is a promising wireless enabling technology because of its low power consumption and potential high bandwidth. Bluetooth is also readily available on many mobile phones. The Bluetooth protocols also provide an easy addressing mechanism for mobile phones that is not achievable with WiFi. To form ad hoc networks using Bluetooth as a connectivity channel, constraints (such as limiting the number of slaves in a piconet to seven and having a master node act as a slave in another piconet) introduced by Bluetooth protocols are taken into consideration. Guérin et. al [27] consider the complexity of topology formation using Bluetooth. They look at how Bluetooth constraints such as the minimum degree for master nodes affects connectivity. The authors distinguish between two types of links between nodes. One is physical, the other a logical Bluetooth link, which exists if Bluetooth establishes a a connection between two nodes. Connectivity therefore exists when the physical links satisfies the degree constraint. Given these constraints, the authors

found that connectivity is possible in a two-dimensional space in which all nodes have the same transmission range if and only if the physical topology is connected. A minimum spanning tree is used to represent this physical topology. In a three-dimensional space, the minimum spanning tree does not fully satisfy the minimum degree constraints.

2.7.1 Forming a Multi-hop Network

The formation of a scatternet is mandatory to develop a connected multi-hop ad hoc network. Figure 2.9 [51] illustrates this setup. A scatternet is an interconnection of piconets. Bluetooth communication typically uses a 79 x 1 MHz bandwidth, divided into 79 equally spaced 1 MHz channels. Bluetooth uses the FHSS-TDD (frequency hopping spread spectrum-time division duplexing) medium access technique to allow up to eight mobile devices to share the same bandwidth [51]. Devices exchange information by establishing a master-slave relationship. After connecting, the devices exchange data by frequency hopping from one channel to the next. In this way transmission collisions are avoided. A master device can connect up to seven slaves to form a piconet.

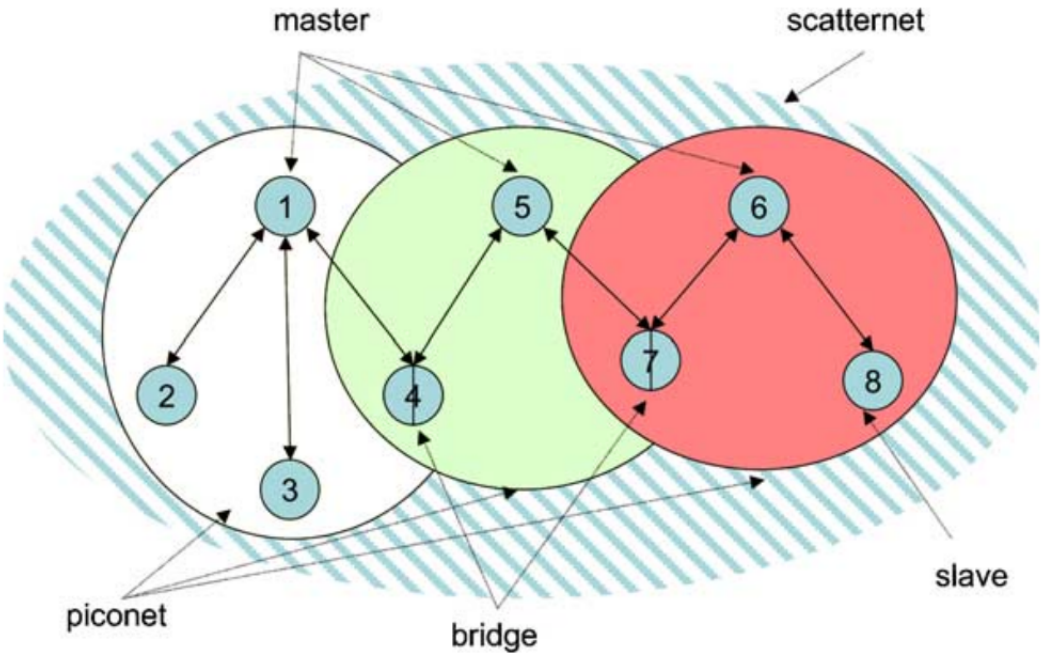


Figure 2.9: An example of a scatternet with three piconets [51]

Several works discuss the formation of multi-hop ad hoc networks using Bluetooth [10, 20, 51, 59, 74, 80]. These works focus on the key characteristic of dynamic topologies in order to optimise scatternet formation. Melodia et al. [51] use a binary matrix to represent the scatternet and evaluate several metrics to determine the performance of the scatternet.

2.7.2 Summary

Bluetooth is a uni-directional technology were discovery of other devices could be affected by the hidden terminal problem experienced in MANETs. Its widespread use, compatibility on many mobile devices, high bandwidth, low power consumption and constant updates makes it easy to use for development purposes. Not much has been done however on searching and file sharing in an overlay Bluetooth ad hoc network that uses a routing protocol.

Chapter 3

Design and Implementation

A Bluetooth ad hoc network consisting of mobile phones connected in a ubiquitous manner is envisaged. Participating mobile phones or nodes will be self-organising and require no central administration. The devices rely on Bluetooth wireless channels to share information. Such a network allows the mobile devices to be self-organising and self-configuring, connecting even devices out of transmission range. Devices use intermediary devices to forward data to devices out of transmission range. Each device should therefore be responsible for maintaining routes to other devices and make routing decisions. Routing in this type of network will be challenging because of the dynamic network topology. Therefore, routing should use the limited resources of the mobile devices efficiently. The routing protocols chosen for information dissemination in this section are able adapt to unpredictable topology changes caused by mobile devices moving further out of transmission range or being switched off.

In order for mobile phones to communicate, a mobile application will be developed that connects the mobile phones to form an ad hoc network. This chapter presents the design and implementation such a mobile application. The chapter commences with an illustration of the type of multi-hop topology to be deployed. The design of the system components is then presented in a top-down fashion: functions at the application layer, the routing layer with design of the routing protocols, and finally the communication layer, which uses Bluetooth for data exchange. Bluetooth communication uses the master/slave relationship to maintain communication between mobile phones. Thus, the application layer implements a user interface that allows mobile devices to execute in either the master or slave role.

This chapter proceeds with the implementation of the design mentioned. Implementation begins with illustrations of the application layer - the sequence diagrams and classes that show the way devices start up and connect. It then proceeds with class diagrams of the different routing protocols. The class diagrams of the three important routing components are presented - route maintenance, route discovery and data requesting; how devices communicate using Bluetooth is also discussed. The chapter concludes with a discussion of the feasibility study conducted to determine whether it is possible to form a multi-hop Bluetooth ad hoc network using mobile phones, and whether routing is possible in such a network. It then addresses the challenges experienced during testing of the prototype mobile application on a test bed.

3.1 Network Topology

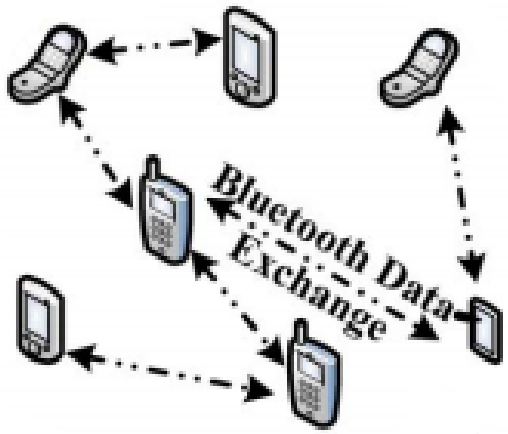


Figure 3.1: Network topology [35]

Figure 3.1 illustrates the mobile phone ad hoc network with nodes communicating in a P2P manner by forming a multi-hop ad hoc network using Bluetooth as a wireless channel. Users with a common interest come together to create this infrastructureless network without any central or fixed administration. Each mobile phone is always aware of its one-hop neighbour. Phones out of transmission range are connected via intermediary nodes creating multi-hop routes.

3.2 System Design

A mobile system is designed that implements three modules, indicated in Figure 3.2. At the application layer, the user interface, data searching, data matching and data merging functionality are implemented; at the routing layer, routing functionality is implemented; at the Bluetooth layer, communication between the mobile phones is implemented.

The system developed is distributed between the mobile devices. When a request is generated or received at the application layer, the application layer functionality is responsible for searching local memory for the requested data. If the data requested are not in local memory, these are forwarded to the routing layer. At the routing layer, a routing protocol makes a decision about where to send the requested data. When a destination address is found, the data packet is forwarded to the Bluetooth communication layer to be exchanged between the mobile devices. Figure 3.2 illustrates this process.

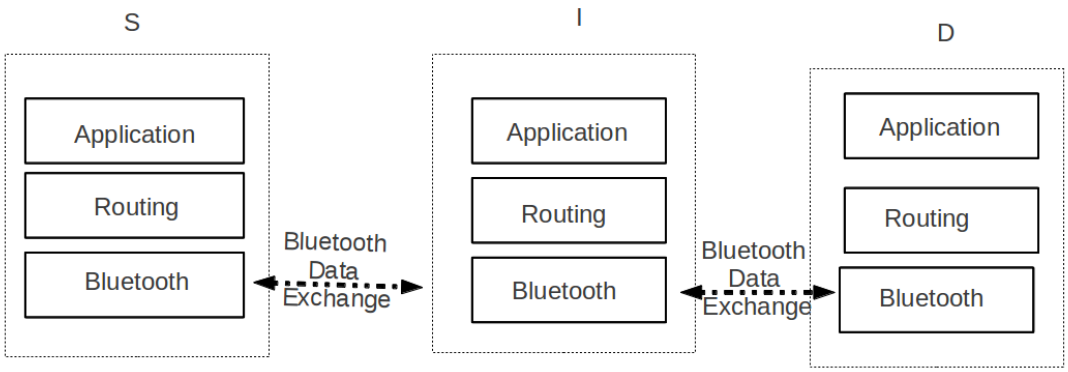


Figure 3.2: System design. S is the source node, I, the intermediary node, and D, the destination node

When a route request is generated at a mobile device, it is wrapped in a route packet at the routing layer and forwarded to surrounding mobile devices using the Bluetooth communication layer. When a route request is received at a mobile device, the application layer receives this request and forwards it to the routing layer. The routing layer will be responsible for routing decisions.

3.2.1 Application

The user interface and data searching or lookup functions are located on the application level.

(a) User interface

The user interface is implemented at the application layer illustrated in Figure 3.2. The user interface allows a user to interact with the application by accepting user input, controlling user input and displaying messages. In the routing component routing protocols are implemented. The communication component is responsible for creating and maintaining Bluetooth connections between mobile devices.

Because the application uses Bluetooth connectivity, it relies on the Bluetooth master/slave feature to create connections. The application can be started either as a Bluetooth master (server) or slave (client). Figure 3.3 shows the process of starting the application in either server or client mode. At the start-up of the application, the Bluetooth feature asks a user to decide in which mode to enter. This leads into either the ClientForm user interface form or the ServerForm form. As the rest of the user interface diagram shows, when a node is started in client mode, it initiates the node discovery process (and is then displayed in the ServiceDiscoveryList). When started in server mode, the application advertises its Bluetooth address and waits for incoming connections before it starts sending and receiving messages.

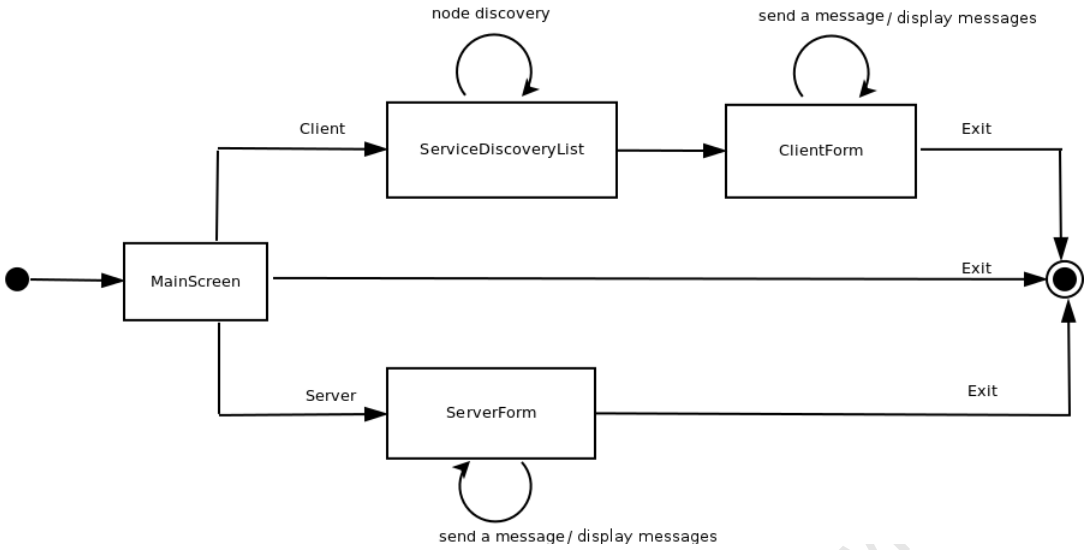


Figure 3.3: User interface design [55]

(b) Data or file searching

The mobile phones store information to be shared with other mobile devices in local memory. Data lookup functionality is designed to search local memory for information from data requests received from neighbouring mobile devices.

3.2.2 Routing

Four routing protocols were chosen for implementation in the Bluetooth ad hoc network. These protocols are broadcasting, on-demand (AODV and DSR), and content-based routing which are presented respectively in this section with pseudocode. The pseudocode illustrates what happens when a node generates a packet, receives a packet during the route discovery, route maintenance and data request. All packets are assigned a TTL parameter. The different packets are abbreviated as follows:

- RREQ: route request
- RREP: route reply
- RERR: route error
- DREQ: data request
- DREP: data reply
- DERR: data error
- HELLO: a hello packet.

3.2.2.1 Broadcasting

In broadcasting, a node will simply propagate a new message forward into the network if it has not seen the message before or if the indicated TTL on the message has not expired. Otherwise the message is discarded.

Route discovery and data request in broadcasting are designed as illustrated in Algorithm 1 and 3. When a node receives a data request or route request message, if it has seen the request before, it discards the message. Otherwise, it stores the request in a buffer and checks to see if the message is intended for itself. If it is, a data reply or route reply is sent back along the same path the request travelled. Otherwise, the request is rebroadcast.

```
1: receive RREQ
2: if (RREQ source IP not in RREQ buffer)
3:         insert in RREQ buffer
4:         write RREQ to outputstream
5: else if (RREQ in RREQ buffer)
6:         ignore RREQ
```

Algorithm 1: Broadcasting a RREQ

Algorithm 2 illustrates what happens at a node when it receives a route reply message. If the reply message is in response to a route request stored in the route request buffer, the route request is removed from the route buffer and the the routing table updated with routing information contained in the route reply message.

```
1: receive RREP
2: if (RREP source IP equal to RREQ source IP in buffer)
3:         remove RREQ buffer
4:         update routing table
```

Algorithm 2: Broadcasting a RREP

```
1: receive DREQ
2: if (DREQ identifier not in buffer)
3:         insert DREQ into buffer
4:         write DREQ to outputstream
5: else
6:         ignore DREQ
```

Algorithm 3: Broadcasting a DREQ

3.2.2.2 On-demand Routing Protocols

On-demand routing protocols are intended for use in mobile ad hoc networks. These protocols are able to adapt quickly to changes in network topologies, require low processing and produce low

control traffic. On-demand protocols establish routes only when demanded. These algorithms are dynamic and offer multi-hop routing whereby intermediary nodes are used to forward requests. These algorithms have three important functions - route discovery, route maintenance and data forwarding. This chapter considers two on-demand protocols, namely AODV and DSR. AODV maintains routing tables with route information indexed by unique sequence numbers to distinguish routes. AODV periodically updates these routes using the route maintenance function. DSR on the other hand, only initiates route discovery when a node wants to forward data in the network. Unlike AODV, DSR does not perform route maintenance. This section presents the design of these two protocols based on the draft specifications by the IETF[1].

(a) Route discovery

Description: Node generates a RREQ

```
1: generate RREQ
2: add RREQ to rreq buffer
3: if (RREQ destination IP is null)
4:         set destination sequence number of RREQ to max
5:         forward the RREQ
```

Algorithm 4: AODV generate RREQ

In Algorithm 4, once a RREQ message is generated by AODV and the destination address is not in the RREQ buffer, it is inserted into the RREQ buffer and then forwarded to surrounding devices.

Description: Node receives a RREQ

```
1: receive RREQ
2: lookup RREQ with destination IP in route table
3: if (RREQ dest. IP not seen before)
4:         update route table
5: if (current IP is destination IP)
6:         generate RREP
7: else if (ttl > 0)
8:         forward RREQ
```

Algorithm 5: AODV receive RREQ

When a node receives a route request, as illustrated in Algorithm 5, the routing table is consulted and if a route in not found, the routing table is updated and the route request broadcast to connected nodes. Otherwise, a route reply is generated. The TTL on the route request message is checked also to see if it has expired.

[1] `http://www.ietf.org/`

Description: Node generates a RREQ

1: generate RREQ
2: insert RREQ into RREQ buffer
3: update routing table/cache with RREQ details
4: forward RREQ to connected neighbours

Algorithm 6: DSR generate RREQ

With Algorithm 6, a RREQ message is first stored locally and used to update the route cache. Finally it is forwarded to connected neighbours.

Description: Node receives a RREQ

1: receive RREQ
2: **if** (route cache contains RREQ destination IP)
3: forward to that destination IP
4: **else**
5: forward RREQ to all connected neighbours

Algorithm 7: DSR receive RREQ

Algorithm 7: Once a RREQ is received, the receiving nodes determine the intended destination and directs the RREQ towards the intended destination. Otherwise, it broadcasts the message to all its connected neighbours.

Because a new network has no prior knowledge of paths that exist, route discovery in AODV and DSR is done via broadcasting. Each node will have a route request table to store new route requests. If the current node is the targeted node, it will send a reply. Otherwise, the route request is rebroadcast.

(b) Route maintenance

Description: Generate a HELLO packet every x milliseconds

1: **while** (true)
2: wait x milliseconds
3: **if** (instance of server connection error raised)
4: generate a RERR
5: **if** (instance of client connection lost raised)
6: generate a RERR
7: **else**
8: send HELLO to backward connected node

Algorithm 8: AODV route maintenance

The aim of route maintenance is to ensure that a link is active between two nodes. In this case, periodic HELLO messages will be sent upwards to neighbouring nodes, otherwise a route error

will be generated, as shown with Algorithm 8.

(c) Data request

Description: Receive DREQ

```
1: reduce DREQ ttl
2: add DREQ to data buffer
3: if (DREQ origin IP equal local address)
4:         generate DREP
5: else if (DREQ origin IP not local address)
6:         if(check local data store)
7:         forward DREQ to destination IP
8:         else
9:                 broadcast the DREQ
10: if (entry lifetime exceeded)
11:         generate route error
12: check buffer for unanswered DREQs
```

Algorithm 9: AODV DREQ

Data requests are not destination address driven: they are forwarded based on content. A node will periodically resend a data request if it does not receive a reply regarding the data request. To prevent loops from occurring, a set number of data request retransmissions is determined for the process. Algorithm 9 shows when AODV receives a DREQ, the TTL is reduced. If the node has the requested data, a data reply is sent back on the reverse path the DREQ travelled. If not, the protocol first checks the current node's local data store for a destination node with matching or the closest matching data being requested.

Description: receive data request

```
1: add route to source record
2: add DREQ to dreq list
3: reduce DREQ ttl
4: if (local memory contains payload in DREQ)
5:         generate DREP
6: else if (local memory is empty & DREQ destination IP is current node)
7:         generate DERR
8: if (local data store has address with data that closely or exactly matches the data)
9:         if (destination in routing table)
10:                forward DREQ to that destination IP
11:        else
12:                broadcast the DREQ
13: check buffer for unanswered DREQs
```

Algorithm 10: DSR DREQ

In algorithm 10, if the receiving node has the required information, it generates a data reply

(DREP). If it does not have the information and does not know where to look, it generates a data error (DERR). If it is able to determine the destination address by resolving which node address might have the required content, it forwards the information to that address. Otherwise, if there are addresses with close matches to the required information, the algorithm broadcasts the request to those nodes' addresses.

Description: Receive DREP
1: receive DREP
2: update local data store with reply content
3: update routing table with DREP addresses
4: remove DREQ from data buffer

Algorithm 11: DREP for AODV and DSR

Algorithm 11 shows how the routing table is updated with data reply information.

3.2.2.3 Content-based Routing

The network consists of nodes that publish content and subscribe to content by expressing interest in specific types of content with a publication or subscription of the form {*type, value*}: *type* represents the type of data and *value* is the actual data request. A broker (master) node is always one hop away from a subscriber or publisher (slave) node, as indicated in Figure 3.4. The broker role is chosen when the application starts.

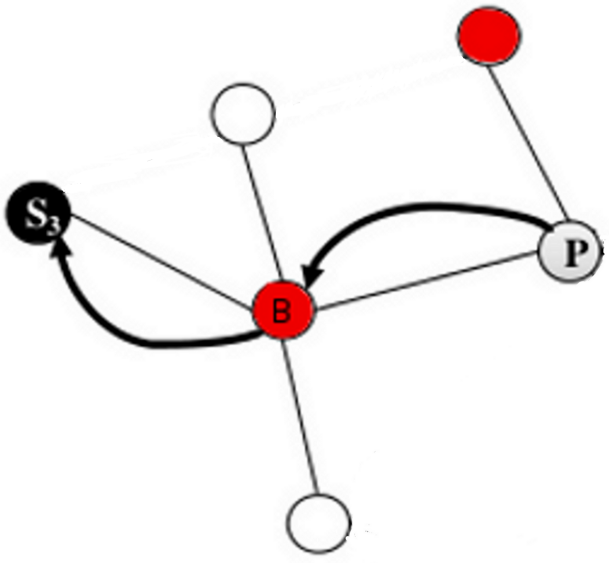


Figure 3.4: B is a broker; S_3, a subscriber; P a publisher node [78]

Description: Receive publication request (PUB)

1: receive PUB
2: **merge** PUB

Algorithm 12: Content-based data request

Nodes will first publish content they have in local memory to the nearest broker. In Algorithm 12, on receipt of a publication, the broker nodes merges the publications at the same type with existing publications in the publication routing table. Otherwise, if the request is already seen, it is discarded.

Description: Generate subscription request
Input: SUB *(type, value)*

1: add sub to sub route table
2: perform merge
3: forward sub to all connected neighbours

Algorithm 13: Content-based generate subscription

Upon generating a subscription request, a node will store the generated request, perform a merge on its data store to find the requested information. If it does not find the requested information, it sends the subscription request to all its connected neighours. This approach is shown in Algorithm 13.

Description: Receive subscription request
Input: SUB *(type, value)*

1: add SUB to subscription table
2: match SUB
3: **if** (match not found){
        forward SUB to connected neighbours
   }
4: **else if** (match found){
        generate DREP
   }

Algorithm 14: Content-based receive subscription

Algorithms 14 shows when a node receives a subscription request, it will search its local data store for the requested information. If it does not have the requested information, it forwards the subscription request to all its connected neighbours. If it does have the requested information, it will generate a DREP back to the node that sent the subscription request.

Description: Merge requests
Input: PUB or SUB with form *(type, value)*

```
1: if (merged set contains PUB.name but not PUB.value){
   insert PUB.value to PUB.name list}
2: else if (PUB.name not in merged set){
   create new PUB entry in merged set }
```

Algorithm 15: Merge function

Description: Match requests
Input: SUB *(type, value)*

```
1: if (SUB.type == type in merged pub set){
       if(SUB.value == value in merged set of type found){
           generate data reply     }
       else{
           broadcast SUB
       }
   }
```

Algorithm 16: Match function

Merging and matching as indicated in Algorithms 15 and 16 are performed at the broker nodes. Merging reduces the sizes of the publication and subscription tables. Matching finds the data or the destination which contains the data or the closest matching data.

Merging is required to remove redundant publications from the routing tables and to reduce memory usage. A merged publication table with have the following form: (t_i, V_i), where t_i is the type of data stored and V_i is the list of values of type t_i. If a message with a type t_i is already seen but with a different value, it is merged with existing messages in the publication table with type t_i. If a subscription with type and value arrives and is matched to a publication request already stored in the publication table, it is discarded.

When a subscription request is received at a node, it is matched (see Algorithm 16) to a publication in a merged publication table. If no match is found, it will be forwarded (similar to broadcasting) to neighbouring nodes. If the request is matched, a reply will be returned to the requesting node.

3.2.3 Bluetooth

Bluetooth imposes a master/slave relationship between mobile devices. One device always acts as the master and the other as the slave device. This simple one-hop network is called a piconet. Multi-hop ad hoc networks are formed with Bluetooth to create what are called scatternets. Scatternets allow one device to act as a master in one piconet, and as a slave in other piconets. In the design, master/slave connections are referred to as server/client connections, as show in Figure 3.6. The server node is responsible for advertising its address to surrounding nodes (clients) and waiting for

incoming connections. The client node performs node discovery. Node discovery is used to set up the ad hoc network by creating multi-hop connections with nodes within and out of transmission range. This process uses Bluetooth's device inquiry and service search processes to discover other nodes.

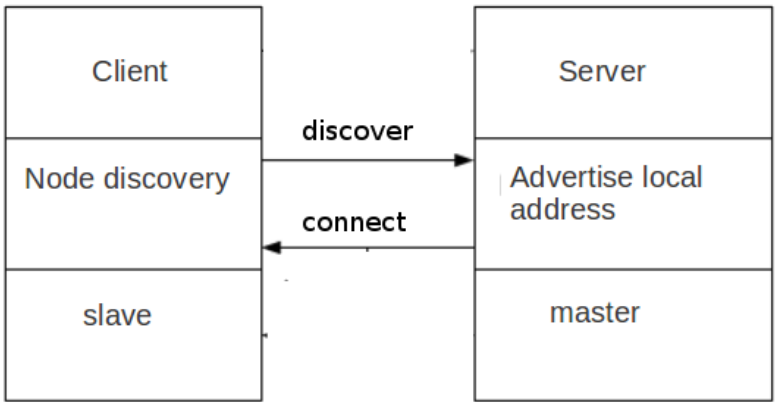


Figure 3.5: Client/server communication

3.3 Implementation

3.3.1 Application

3.3.1.1 Peer-to-Peer Communication

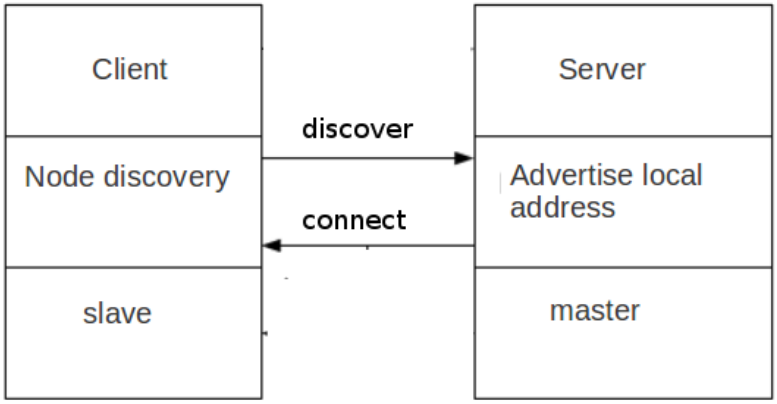


Figure 3.6: Client/Server communication

Bluetooth's master/slave feature is implemented as a client/server relationship between communicating nodes. As illustrated in 3.6, at the application layer, the client is the slave node. The server

node plays the Bluetooth master role. The server and client nodes share the same functionality and communicate in a P2P manner. Each node is able to:

1. Perform route discovery.
2. Perform route maintenance.
3. Send and receive requests.
4. Perform data lookup.

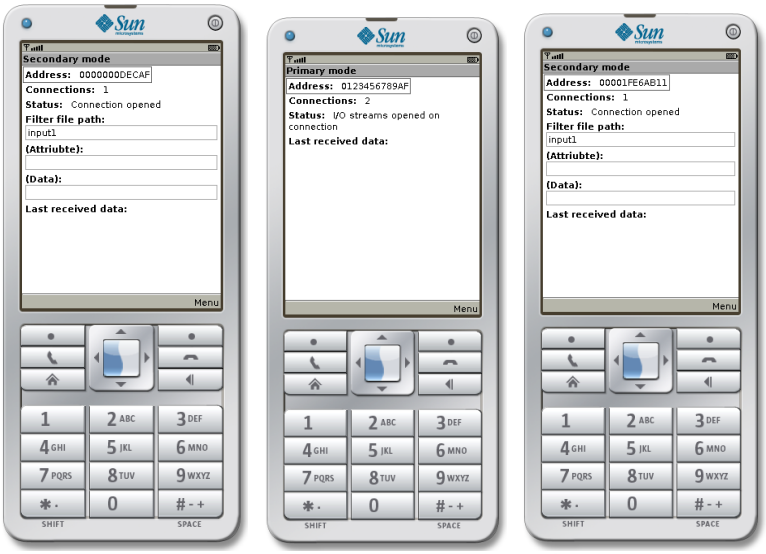
(a) Client node equivalent to slave node of piconet (b) Server node equivalent to master node of piconet

Figure 3.7: Emulators at start up

Figures 3.7(a) and 3.7(b) show the emulators at start-up and what modes can be chosen. In Figure 3.8(b) the master mode was chosen, while slave modes in Figures 3.8(a) and 3.8(c) connect to the server client. A client node connects to a server by initiating the route discovery sequence shown in Figure 3.11.

3.3.1.2 The User interface

J2ME midlets were used to implement the user interface. The midlet class controls interface operation, displays button controls and is responsible for terminating the application. Figure 3.9 shows the main classes of the user interface. Two display forms, namely ServerForm and ClientForm, are administered by the OppNET midlet. The ServerForm was implemented for nodes in server mode. The ClientForm was used for the client mode.



(a) Node A (b) Node B (c) Node C

Figure 3.8: Emulators when connected: Node A connects to Node B; and Node C connects to Node B.

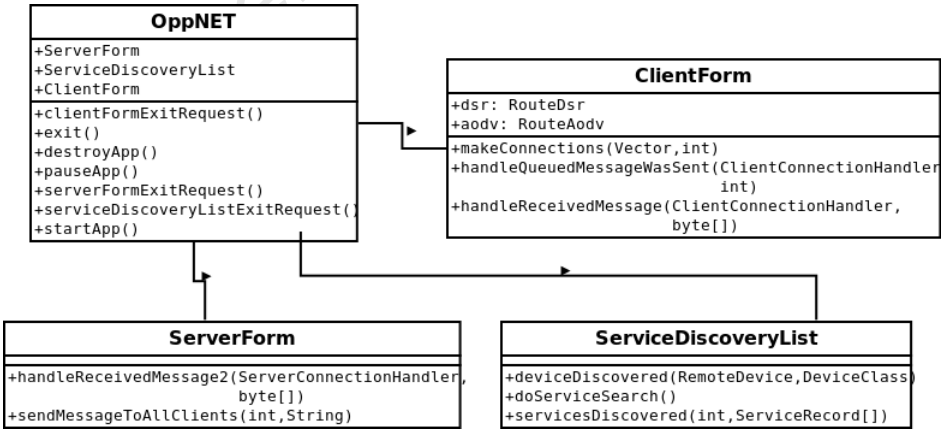


Figure 3.9: User interface classes

3.3.1.3 Platform

The mobile application was developed using the *Java 2 Platform Micro Edition* (J2ME) platform that uses the Mobile Information Device Profile (MIDP) 2.1 and the Connected Limited Device Configuration (CLDC) 1.1 configurations. Four versions of the application are implemented, one for each of the routing protocols discussed in the previous chapter.

3.3.2 Routing

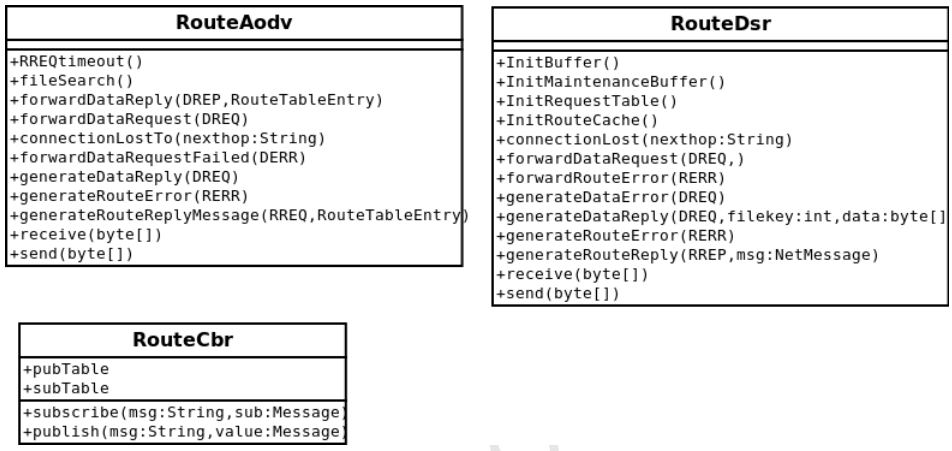


Figure 3.10: Routing classes

Each routing protocol was implemented as a separate class, as indicated in Figure 3.10. Broadcasting used the send and receive functions in the ClientConnectionHandler and ServerConnectionHandler class to propagate requests into the network.

3.3.2.1 Route discovery

Route discovery began with a node in client mode, as shown in Figure 3.11. The ServiceDiscovery process was started and discovered all nodes in server mode. A list of discovered devices is returned. From the list, the client node decided which nodes to connect to.

3.3.2.2 Route maintenance

Figure 3.12 shows the route maintenance sequence implemented for AODV. After route discovery, a periodic HELLO message was sent to the upward one-hop neighbour to ensure that a link was still active.

3.3.2.3 Data requesting

In Figure 3.13, Node A sends a message to Node B, which forwards it to Node C.

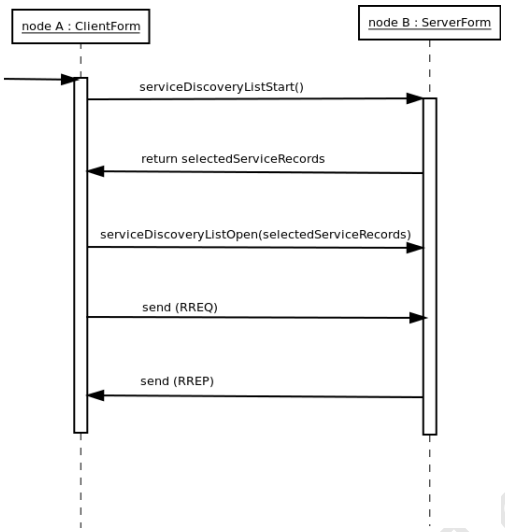


Figure 3.11: Route discovery sequence

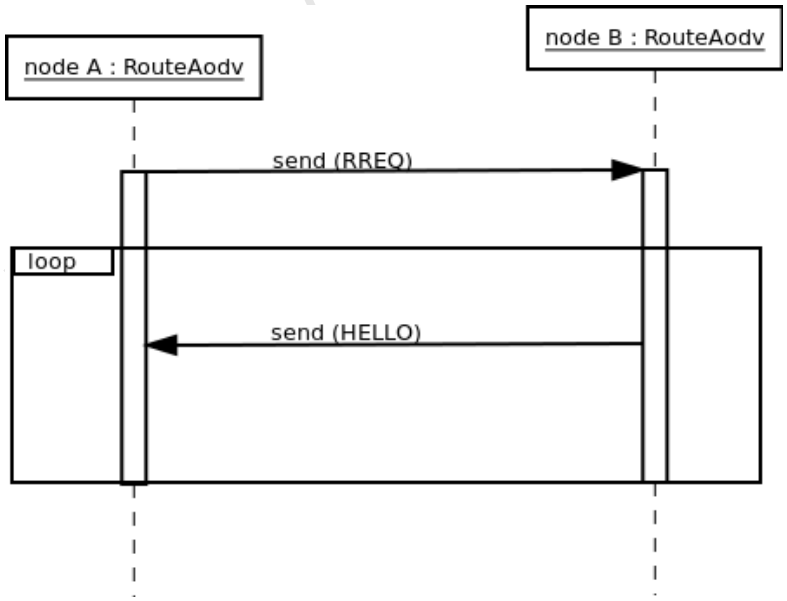
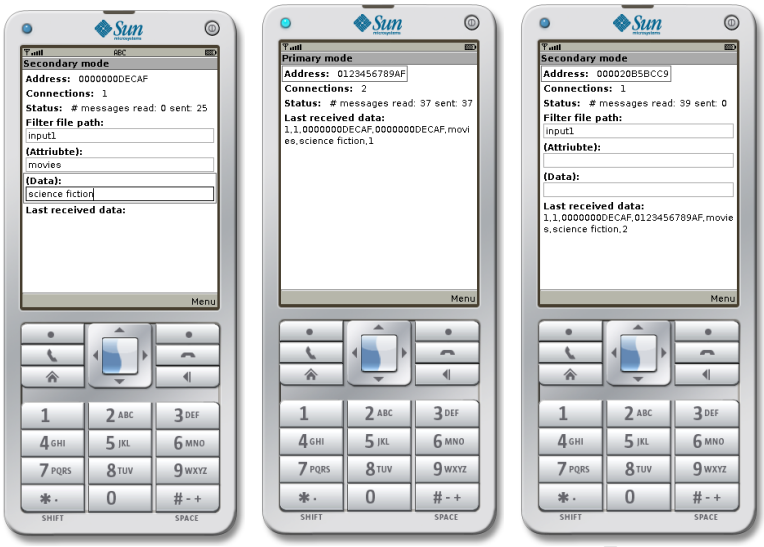


Figure 3.12: Route maintenance sequence



(a) Node A (b) Node B (c) Node C

Figure 3.13: Emulators forwarding a data request

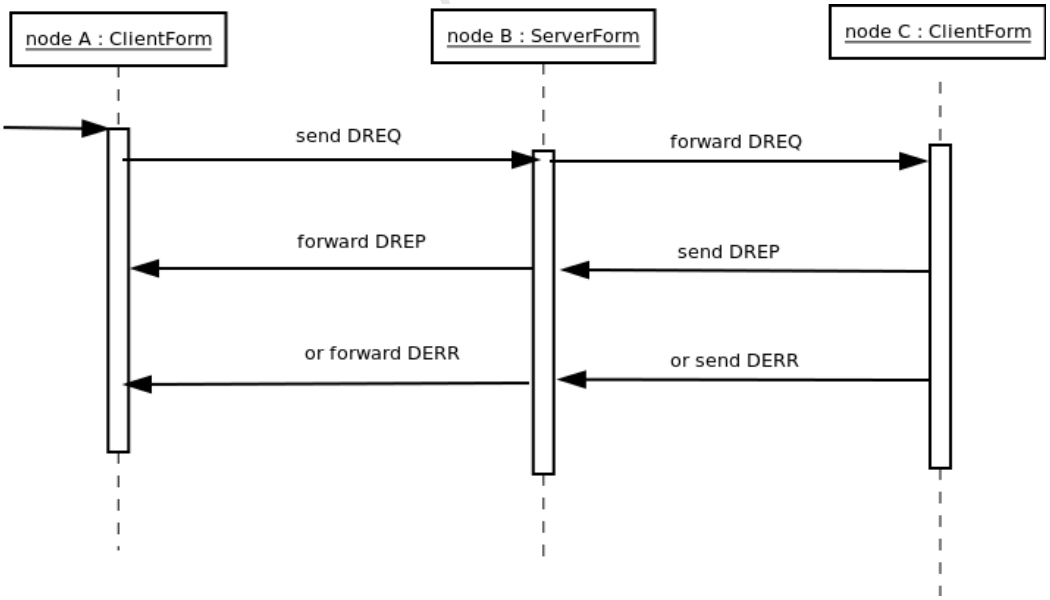


Figure 3.14: Data request sequence

A data request initiated sent out a DREQ message (Figure 3.14). Two replies are expected back by the node in client mode: a data reply indicating that the neighbouring node had found the data or a data error indicating that the neighbouring nodes had failed to find the requested information. Figure 3.14 shows what the interfaces look like after Node A in Figure 3.13(a) sends out a data request. Node B in Figure 3.13(b) receives the data request and then forwards it to Node C in Figure 3.13(c).

3.3.3 Bluetooth

The Bluetooth component is implemented as shown in Figure 3.15. The class implementations of the master/slave or server/client relationships are shown in the figure. The ClientConnectionHandler class implemented the send and receive functions on the client side. The ServerConnectionHandler was responsible for sending and receiving data on the server side. The ServerConnectionHandler class acted as a Bluetooth master and advertised its address and waited for connections from the ClientConnectionHandler class.

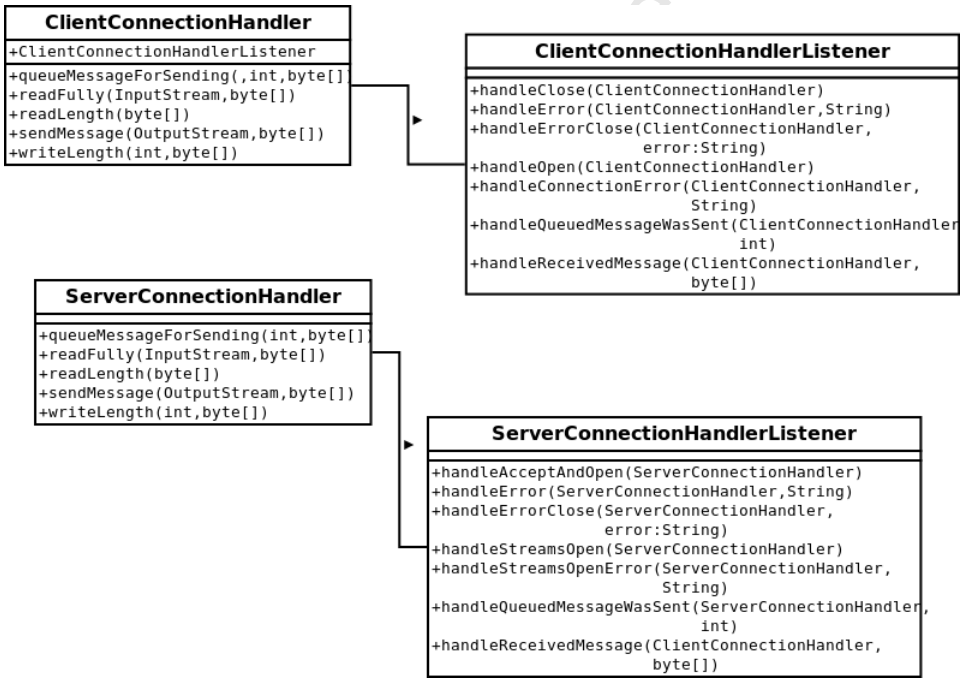


Figure 3.15: Communication classes

3.4 Summary

This chapter presented the design and implementation of the user interface, the communication between nodes and the algorithms to be used in the sending and receiving of packets during the

route discovery, route maintenance and data request phases of the routing protocols. The design of communication between nodes uses the Bluetooth master/slave relationship to allow the nodes to start up as either a server or client. The Bluetooth master/slave feature forces the design to be divided into two modes - a server and client mode. The master relationship is associated with the server mode. The server mode is responsible for advertising its address and waiting for incoming connections. The client mode initiates communication with the node in server mode. This relationship is imposed by the Bluetooth protocol standard.

Implementation of the mobile application is shown using class diagrams and sequence diagrams. These diagrams show how route discovery, route maintenance and data requesting were implemented. The chapter also discussed the Bluetooth master-slave feature that forced the mobile application be implemented in two modes. That is, the mobile node took on the role of either a master node or client node when it started up.

Chapter 4

System Evaluation

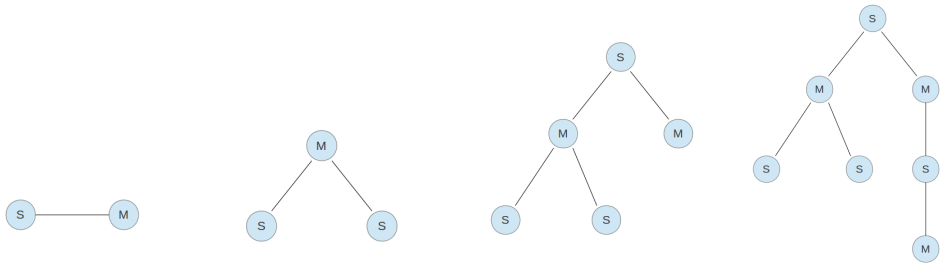
Evaluation of the performance of routing in the Bluetooth ad hoc network is presented in this chapter. The chapter begins by presenting the experiment setup along with the topologies used to create the Bluetooth ad hoc network. This chapter discusses the feasibility study conducted to implement a Bluetooth ad hoc network with nodes communicating in a P2P. Once the feasibility study has been concluded, an evaluation of each routing protocol performance according to defined routing metrics is given. The chapter then does a comparison of the different routing protocols according to the routing metrics. The chapter finally concludes with a discussion to determine which implemented routing protocols are suitable for the Bluetooth ad hoc network.

4.1 Experiment Setup

This section describes the Bluetooth ad hoc network topologies experimented with, the messages created for querying, how the experiments were carried out and finally the data collected and the analysis of those data. This section further defines the routing metrics used to determine performance of the routing protocols.

4.1.1 Physical Topology

Four physical topologies for the Bluetooth ad hoc network were experimented with, as shown in Figure 4.1. Each topology has a fixed number of nodes, n, and the number of hops satisfies the relationship ($\sqrt{nlogn}$). The number of nodes, also referred to as the *network size*, is used as the independent variable. Each topology requires that participating nodes are started either in the master mode or slave mode. This is a feature of the Bluetooth protocol, which requires that one node be the master of communication. Figure Figure 4.1 shows the topologies used during testing. In Figure 4.1(b), the slave node S initiates communication with the master node M by sending out a route query, and maintains communication with node M by issuing user requests.



(a) two-node network (b) three-node network (c) five-node network (d) eight-node network

Figure 4.1: Network topologies (M = master, S = slave). Each circle represents a mobile phone.

4.1.2 Message Generation

During route discovery, a route request message is transmitted by the node, which initiates a connection with its neighbours. Upon receiving a route request message, it initiates a route reply message. During data request transmission, a node sends out a data request of the form {*type, value*}. The node that initiates communication is usually the one that sends the first data request. To collect, data transmission is emulated with twenty trial runs. During each trial run, a random node generates fifty random data requests, which are transmitted throughout the network.

4.1.3 Experimentation

Experimentation was carried out using emulation instead of simulation. Emulation was executed in a controlled environment in order to obtain comparable results. The test bed used in the feasibility study highlighted some challenges with that approach. The test bed was difficult to monitor. Some of the challenges experienced were connections between devices being interrupted by other devices wanting to connect to the same device. Once connections were disrupted, re-establishing these connections required restarting the testbed experiments according to the topologies shown in Figure 4.1. Wireless network simulators use many mobility models, such as random waypoint mobility, which do not realistically model the dynamicity of the Bluetooth ad hoc network described in this research. In addition, simulation requires that the real world be modelled instead of observing behaviour of the mobile application developed. However, emulation is an exhaustive process that requires user monitoring. For this reason, four physical topologies were chosen (see Figure 4.1) to represent from as small a network as possible to a large network consisting of eight nodes.
The experiment was carried out as follows:

1. Start the emulators (equal to the network size).
2. Start the Bluetooth device discovery process.
3. Connect the devices to form a multi-hop network as shown in Figure 4.1.
4. Send and receive routing packets.

5. To simulate network disruptions, an emulator in master mode is switched off during the experiments.

6. Each message generated in the network has a TTL in milliseconds. Messages are also kept in the RREQ and DREQ buffers respectively, until they expire or a new message is generated.

7. Network behaviour is observed for four minutes, i.e. enough time is given for sending and receiving messages and simulating network disruptions.

4.1.3.1 Data Collected

source/prevhop	nexthop	messageid	messagetype	packetsize(bytes)	hopcount	entrytime(ms)	action
0000000DECAF	0123456789AF	0	null	24	0	1354546599842	orig.rreq
0000000DECAF	0123456789AF	0	null	20	0	1354546599952	orig.rrep
0000000DECAF	0123456789AF	0	null	24	1	1354546599955	recv.rreq
0000000DECAF	0123456789AF	0	null	20	1	1354546599976	recv.rrep
0123456789AF	000006298E1B	0	null	24	0	1354546606023	orig.rreq
0123456789AF	000006298E1B	0	null	20	0	1354546606118	orig.rrep
0123456789AF	000006298E1B	0	null	24	1	1354546606122	recv.rreq
0123456789AF	000006298E1B	0	null	20	1	1354546606140	recv.rrep
0123456789AF		1	null	20	1	1354546609978	orig.hello
0123456789AF		1	null	34	1	1354546609985	recv.hello
000006298E1B	0000E8DC4036	0	null	24	0	1354546613335	orig.rreq
000006298E1B	0000E8DC4036	0	null	20	0	1354546613437	orig.rrep
000006298E1B	0000E8DC4036	0	null	24	1	1354546613441	recv.rreq
000006298E1B	0000E8DC4036	0	null	20	1	1354546613456	recv.rrep
000006298E1B		1	null	20	1	1354546616139	orig.hello
000006298E1B		1	null	34	1	1354546616142	recv.hello
0000E8DC4036	00005822C132	0	null	24	0	1354546618648	orig.rreq
0000E8DC4036	00005822C132	0	null	20	0	1354546618749	orig.rrep
0000E8DC4036	00005822C132	0	null	24	1	1354546618752	recv.rreq
0000E8DC4036	00005822C132	0	null	20	1	1354546618773	recv.rrep
0123456789AF		1	null	20	1	1354546619979	orig.hello
0123456789AF		1	null	34	1	1354546619987	recv.hello

Figure 4.2: Data collected

Figure 4.2 shows a sample of the dataset collected for each network topology for each routing protocol. In Appendix 5, the averaged data used to plot the graphs in this document are shown.

4.1.3.2 Data Analysis

1. Python scripts were used to calculate the average data the different metrics.

2. Data for the independent variables number of nodes and hop count is gathered.

3. Data were graphically represented using box plots. Each data set is represented by a box plot drawn from the first quartile (25%) to the third quartile (75%), and the median is indicated by a line through the box. Whiskers from both ends of the box indicate the minimum and maximum values of the data set. Data points outside the range of the box and whiskers are considered outliers and are indicated with a cross. The mean is shown by a small square. In Appendix 5, the averaged and sum figures obtained over the iterations are shown.

4. The mean is used to describe the behaviour of the performance metrics during data analysis. A description of the general trend of the data is first given, followed by an explanation of outliers and the variation between the box plots.

5. To aid with the interpretation the results, a table of statistics containing the median, mean, interquartile range and decile range is presented for each box plot.

4.2 Peer-to-Peer Feasibility

The aim of the feasibility study was determine how nodes can communicate in a P2P manner using Bluetooth. To establish true P2P communication, the initial aim was that the nodes should be able to advertise their local address to neighbouring nodes and discover neighbouring nodes at the same time. The first prototype implemented these two functions using a multi-threaded design. The prototype mobile application ran two threads - one advertising its local address and the other waiting for incoming connections from surrounding nodes.

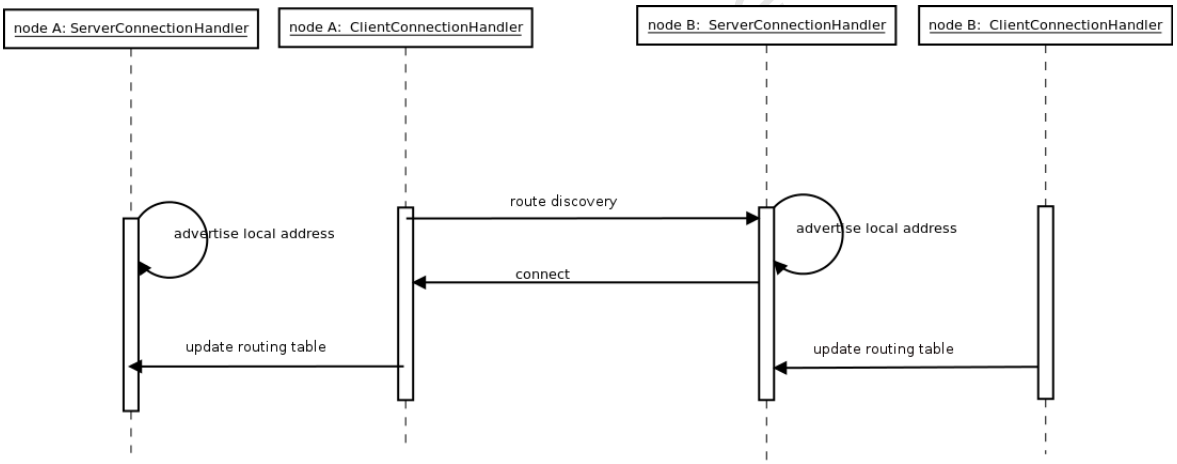


Figure 4.3: First prototype implementation

As illustrated in Figure 4.3, each node ran two threads - the ServerConnectionHandler thread that advertised the node's local address then waited for incoming connections; and the ClientConnectionHandler thread that performed node discovery to find surrounding neighbouring node addresses. When a node ran the prototype, it began by advertising its local address. When route discovery was initiated, node A connected to node B's waiting ServerConnectionHandler thread. When a connection was established, node A updated its routing table with node B's address.

The prototype tested how AODV and DSR routed data through the Bluetooth ad hoc network. The prototype was tested on a test bed of five Nokia N96 mobile phones with 128 MB RAM, and expandable memory up to 16 GB with an SD card. The devices had Bluetooth version 2.0 and

Java MIDP 2.1.

The data obtained were insufficient and showed much variation due to human interaction during testing; some nodes were unable to receive data packets from neighbouring nodes. The second prototype separated implemented only one thread in each node - a node could choose to advertise its local address and wait for incoming connections while another node initiated route discovery. This approach was more suitable to implement with the Bluetooth master/slave relationship imposed by the Bluetooth protocol. The route discovery process was assigned to the Bluetooth slave node while the local node address advertising was assigned to the master node. Therefore, at the data link layer, the nodes implemented the Bluetooth master/slave feature, but nodes had equal or P2P roles at the application layer. Further testing of the new prototype was done in a controlled environment in order to obtain more data.

4.3 Routing Protocols

This section answers the research question: How do existing routing protocols perform in the multi-hop Bluetooth ad hoc network? The results of routing protocols implemented are presented in this section. Firstly, a definition of the routing metrics evaluated is given, followed by box plots for the different routing protocols. This section studies how each routing protocol performs.

4.3.1 Performance Metrics

This section defines the routing performance metrics chosen to evaluate routing performance.

4.3.1.1 Total Traffic

Total traffic, T_T, refers to the number of messages, msg, that pass through an active link and are received at each node. Traffic includes periodic update messages, route requests, route replies, route error messages, data requests, data replies and data error packets. Total traffic is measured in bytes and can be used to interpret how much total power will be used in the network by the mobile phones. Total traffic is calculated as follows:

$$T_T = \Sigma msg. \tag{4.1}$$

The aim of this metric is to reflect how the routing protocol affects mobile phones in total when the network size and number of messages in the network increase. This is an important metric, for example, it could influence the duration or battery lifetime of the mobile phones.

4.3.1.2 Data Traffic

Data traffic, T_D, refers to successfully received data messages, msg measured in bytes, received at each node. It excludes control messages. Data traffic is represented as follows:

$$T_D = \Sigma msg. \tag{4.2}$$

The aim of data traffic is to show the effectiveness of the routing protocols in delivering only data packets.

4.3.1.3 Control Traffic

Control traffic refers to the difference between the total traffic, that is T_T, and data traffic, T_D, in the entire network. Control traffic includes all route discovery and route maintenance messages.

$$T_C = T_T - T_D. \tag{4.3}$$

The aim of this metric is to determine how much total traffic is due to control traffic, and which routing protocols transmit to as many nodes as possible with little control traffic.

4.3.1.4 Delay

Delay, D_T, is the average amount of time between when a message is sent from a source node and received at a destination node, averaged over the number of times, m, the source node generates messages intended for that specific destination. It consists of processing delay at each node and transmission delay. Average delay is calculated as follows:

$$D = \Sigma \frac{t_{recv} - t_{sent}}{m}. \tag{4.4}$$

The aim of this metric is to determine which routing protocol is faster at transmitting messages through the network.

4.3.1.5 Convergence Time

In the Bluetooth ad hoc network, convergence time refers to the time taken to establish a stable network topology. Convergence time is determined by measuring the time it takes to send a route request from a source node to the time it was received at its destination, that is:

$$C = t_{recv} - t_{sent}. \tag{4.5}$$

The aim of this metric is to determine how quickly a network adapts to changes.

4.3.1.6 Positive Response

The positive response (equation 4.6) is the ratio of the number of data reply messages, $rmsg_{recv}$ in bytes, successfully received at the source node to the number of data request messages sent, $dmsg_{sent}$ in bytes, by the same source node. A reply message is either a successful data reply or a

data error saying the data request could not be fulfilled:

$$DR = \frac{rmsg_{recv}}{dmsg_{sent}} * 100. \tag{4.6}$$

The aim was to determine how well routing protocols deliver data requests to destinations and replies to sources.

4.3.2 Broadcasting

4.3.2.1 Total Traffic

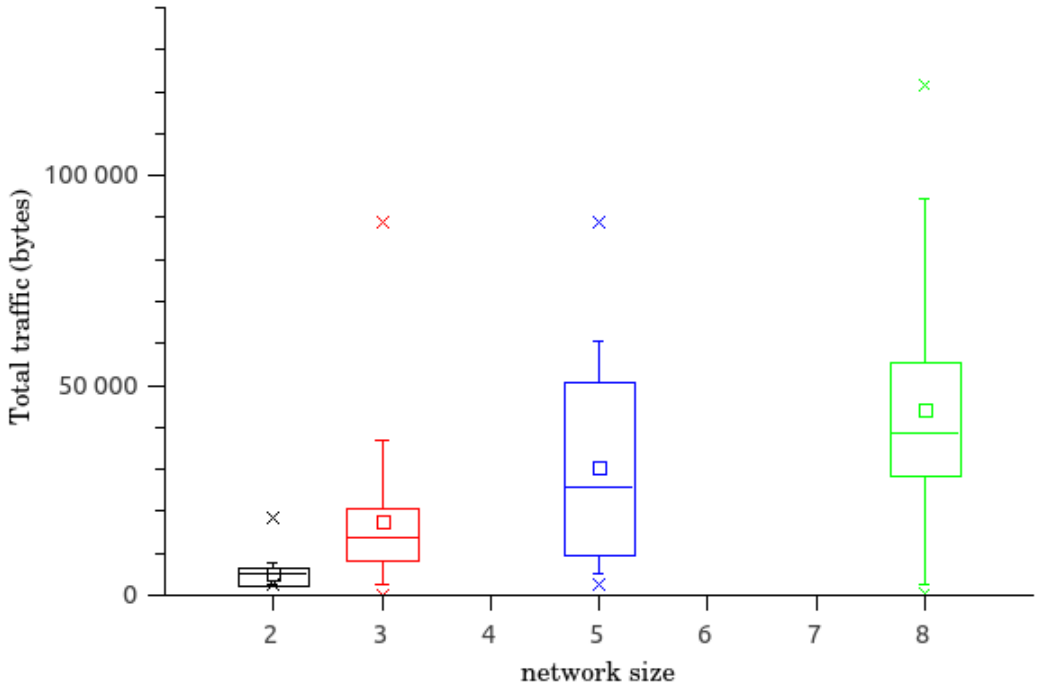


Figure 4.4: Broadcasting total traffic

Network size	1st decile	1st quartile	Median	3rd quartile	9th decile	Mean
2	2 402,70	2 644,75	5 050,00	6 588,00	6 915,60	5 295,90
3	4 596,10	8 437,00	13 668,50	20 802,25	27 636,40	17 675,90
5	8 031,20	9 817,50	25 765,50	50 607,00	57 535,70	30 701,65
8	7 765,90	28 638,75	38 808,00	55 606,75	89 243,70	44 073,25

Table 4.1: Broadcasting total traffic statistics

In a comparison between the network sizes (Figure 4.4), the mean values show that as the network

size increased, traffic increased. This is because as the number of nodes increased, more packets were broadcast through the network. More messages were transmitted through the network as more links became available. The interquartile range shows that total traffic increased significantly as network size increased. Outliers equal to zero represent nodes that were switched off during transmission - used to emulate user behaviour and nodes moving out of transmission range.

4.3.2.2 Data Traffic

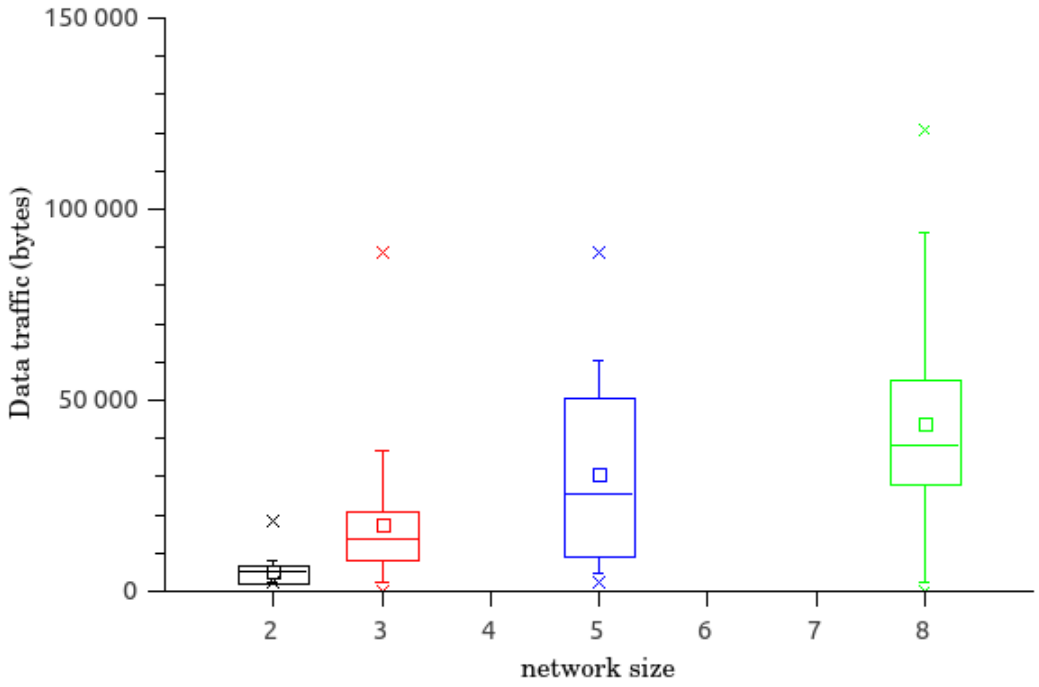


Figure 4.5: Broadcasting data traffic

Network size	1st decile	1st quartile	Median	3rd quartile	9th decile	Mean
2	1 201,20	2 068,75	2 638,00	5 504,00	6 578,40	5 233,90
3	1 616,90	2 440,25	8 455,50	18 104,75	25 777,50	17 551,90
5	3 022,80	5 464,50	12 727,00	23 850,00	39 279,20	30 453,65
8	5 717,40	9 451,00	26 198,50	48 397,50	69 350,40	43 639,25

Table 4.2: Broadcasting data traffic statistics

Figure 4.5 shows that data traffic increased as the network size increased. As more nodes were added to the network, more packets were rebroadcast. Data traffic increased significantly, as indicated by the differences in the interquartile ranges between the different network sizes. The zero outliers represent node dynamicity, that is, nodes moving out of transmission range or being switched off.

4.3.2.3 Control Traffic

Network size	Control traffic (bytes)
2	62
3	124
5	248
8	434

Table 4.3: Broadcasting control traffic

Control traffic increased significantly as the network size increased. The control traffic was fixed in broadcasting because the routing protocol generated only a specific number of control packets at the beginning of the route discovery process to discover surrounding nodes. No route maintenance was initiated throughout broadcasting.

4.3.2.4 Delay

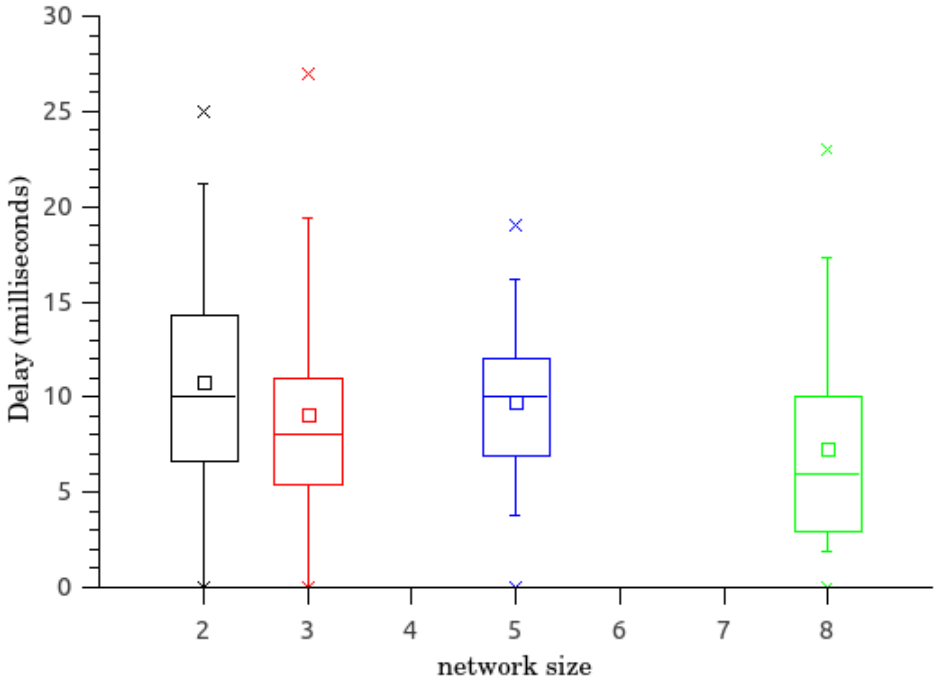


Figure 4.6: Broadcasting delay

As seen in Figure 4.6, average delay decreased gradually as the network size increased. The number of nodes increased connectivity between nodes and thus decreased delay. Because the Bluetooth ad hoc network sees some network disruptions (nodes being switched off) and every node is aware

Network size	1st decile	1st quartile	Median	3rd quartile	9th decile	Mean
2	3,60	6,75	10	14,25	19,20	11
3	1,80	5,50	8	11	17,20	9
5	5,80	7	10	12	15,10	10
8	2	3	6	10	11,60	7

Table 4.4: Broadcasting delay statistics

and connected to surrounding devices, delay decreased.

4.3.2.5 Convergence Time

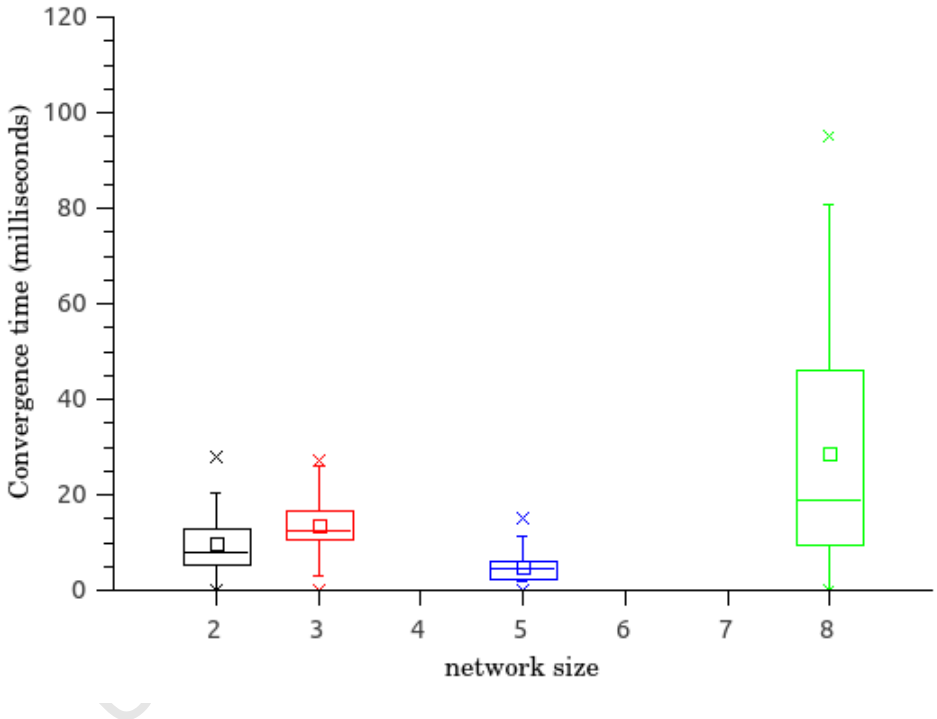


Figure 4.7: Broadcasting convergence time

Convergence time increased as the network size increased. As the network size increased, more unknown destinations were added to the network, which increased the delay in discovering these new destinations. It took longer to update the routing tables with the new routes. The broadcasting convergence time for network size 5 was lowest. At this stage, nodes only established connections with the closests devices, forming an ad hoc network more quickly.

Table 4.5: Broadcasting convergence statistics

Network size	1st decile	1st quartile	Median	3rd quartile	9th decile	Mean
2	4,50	5,75	8	13	19,10	10
3	5,70	11	12,50	16,50	22,40	14
5	2	2,75	4,50	6	7,40	5
8	1,80	10	19	46	55,70	29

4.3.2.6 Positive Response

Network size	Positive Response (%)
2	9,42
3	10,83
5	26,12
8	16,52

Table 4.6: Broadcasting positive response

Broadcasting had a poor positive response because there was no follow-up procedure to resend data requests that were simply lost with no replies generated. The increase in positive response could be due to the new nodes, which had no information about where content was in the network.

4.3.3 AODV

4.3.3.1 Total Traffic

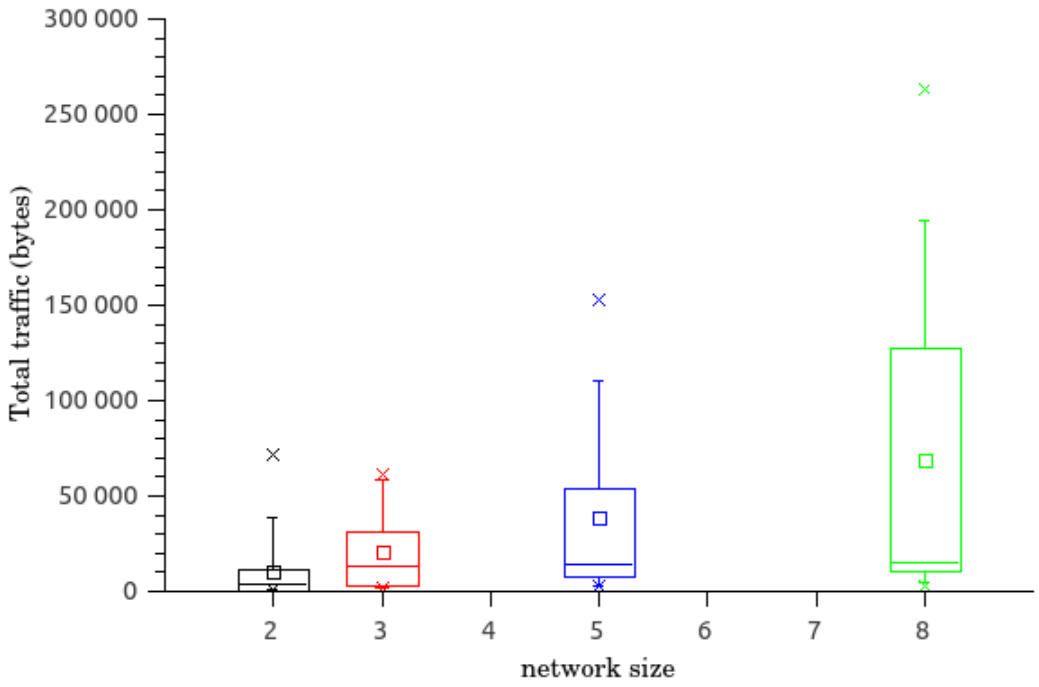


Figure 4.8: AODV total traffic

Network size	1st decile	1st quartile	Median	3rd quartile	9th decile	Mean
2	736,00	1 104,25	3 416,00	11 115,00	14 551,80	10 164,60
3	3 027,80	3 833,50	13 661,00	30 886,75	56 704,00	21 056,55
5	5 265,50	8 458,25	14 203,00	53 646,50	100 775,20	38 842,20
8	6 002,40	11 620,50	15 146,00	127 756,50	189 249,80	69 038,35

Table 4.7: AODV total traffic statistics

AODV total traffic increased as the network size increased. This happened because as the number of nodes increased, with more new destinations in the AODV routing tables, traffic increased. Total traffic increased significantly, as calculated from the interquartile range differences between the different network sizes. This significant increase in total traffic was due to the periodic HELLO messages, which were configured to update every five seconds in the network. Zero outliers were caused by nodes moving out of transmission range or being switched off.

4.3.3.2 Data Traffic

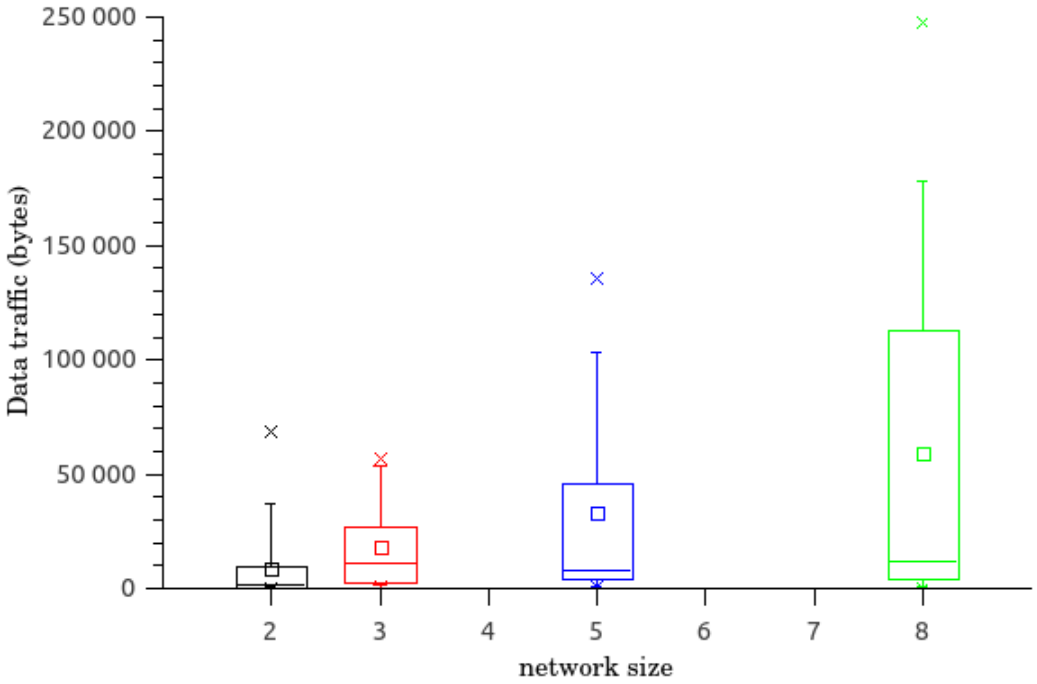


Figure 4.9: AODV

Network size	1st decile	1st quartile	Median	3rd quartile	9th decile	Mean
2	451,60	838,00	1 800,00	9 407,00	12 892,40	8 924,10
3	2 459,60	2 788,50	10 654,00	27 011,75	53 326,20	18 433,45
5	1 250,30	4 957,25	7 871,00	45 589,00	94 285,20	33 062,6
8	3 088,90	4 344,75	11 955,00	112 553,00	161 246,80	58 926,85

Table 4.8: AODV data traffic statistics

In Figure 4.9, data traffic increased steadily as the network size increased, because more nodes were transmitting more data packets as the network size increased. The zero outliers were caused by nodes moving out of transmission range or being switched off.

4.3.3.3 Control Traffic

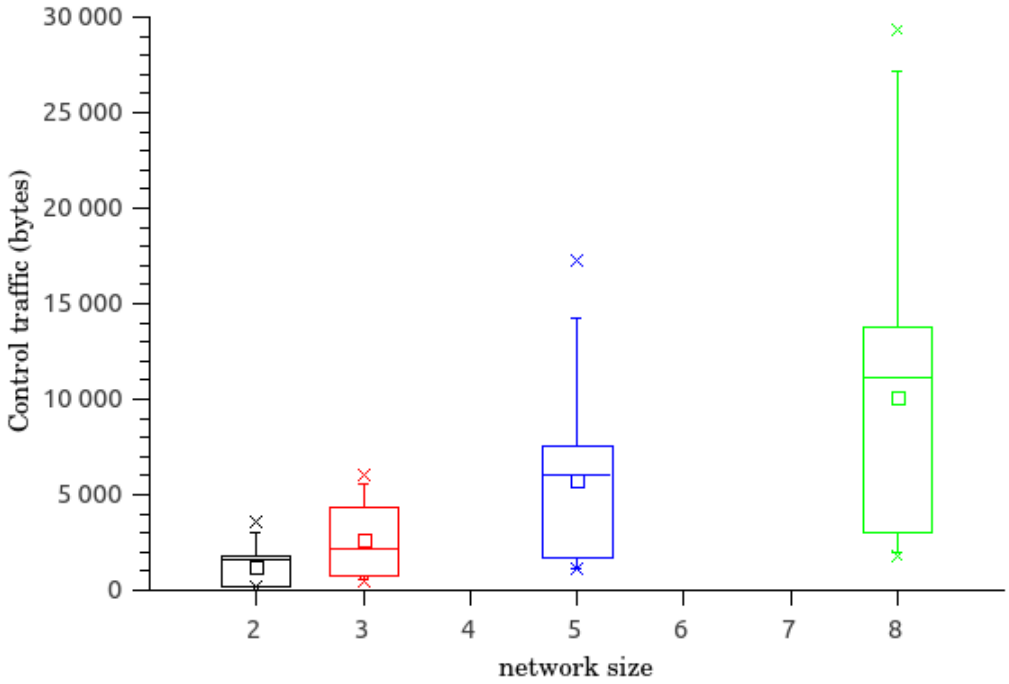


Figure 4.10: AODV control traffic

Network size	1st decile	1st quartile	Median	3rd quartile	9th decile	Mean
2	196,00	245,00	1 627,00	1 789,00	2 368,00	1 240,50
3	620,60	879,50	2 174,00	4 293,50	5 219,80	2 623,10
5	1 279,20	1 792,00	5 991,00	7 527,50	11 107,80	5 779,60
8	2 136,40	3 122,50	11 172,00	13 741,50	17 141,60	10 111,50

Table 4.9: AODV control traffic statistics

AODV control traffic increased gradually as the network size increased. More control traffic was generated in AODV because of the number of periodic HELLO packets generated every five seconds in the network resulting, in a gradual increase in total traffic. However, the frequence of HELLO messages is configurable in the code.

4.3.3.4 Delay

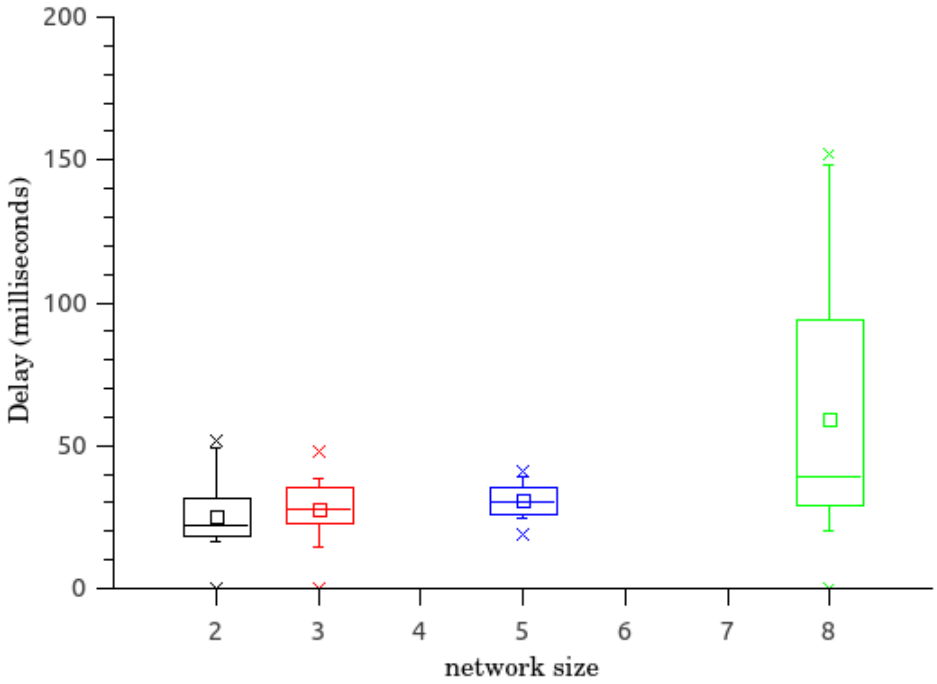


Figure 4.11: AODV delay

Network size	1st decile	1st quartile	Median	3rd quartile	9th decile	Mean
2	17	18,75	22	31,25	37,30	25
3	18,60	23,50	28	35,50	37,10	28
5	25	26,75	30	35,25	38,10	31
8	25,50	29,75	39	94	110,20	59

Table 4.10: AODV delay statistics

Delay increased slightly as the network size increased. The increase in delay was also caused by the processing time required to determine the next route from the routing tables.

4.3.3.5 Convergence Time

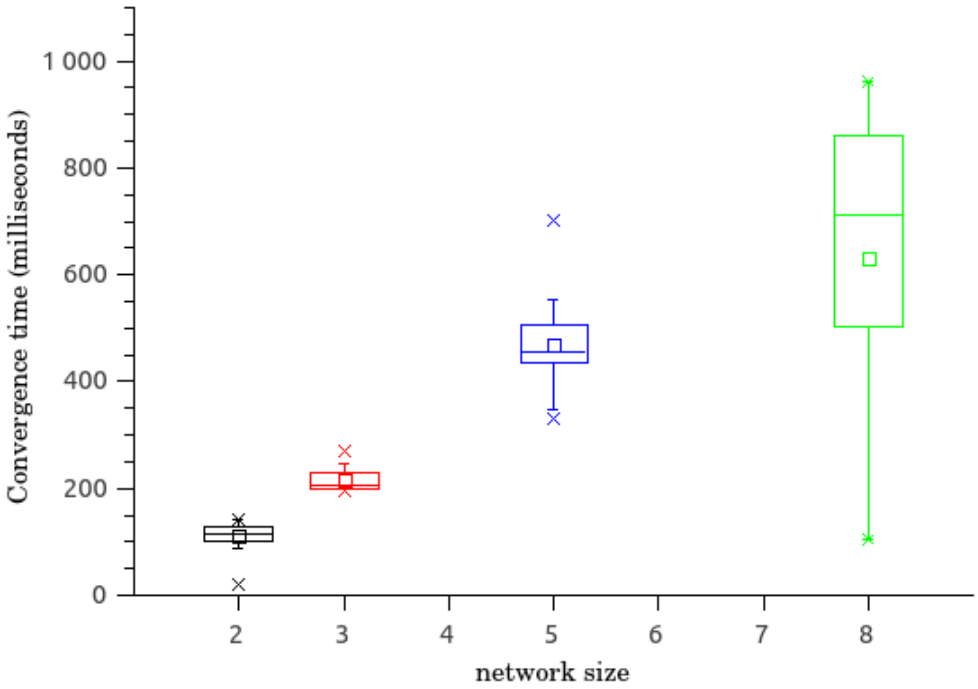


Figure 4.12: AODV convergence time

Table 4.11: AODV convergence time statistics

Network size	1st decile	1st quartile	Median	3rd quartile	9th decile	Mean
2	100,20	106	115	128,25	138,30	113
3	199	202,75	207,50	230,25	244	217
5	411,80	437,50	455	504,75	528,70	471
8	105,80	506,25	713	861,25	937,60	631

Convergence time increased significantly as the network size increased, because a node started off with no routing information about surrounding nodes. Route discovery was initiated, which took time to discover neighbouring nodes.

4.3.3.6 Positive Response

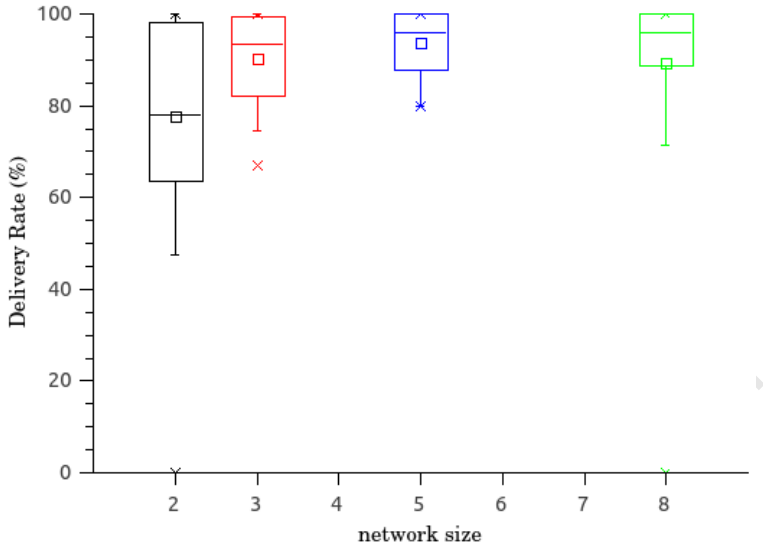


Figure 4.13: AODV positive response

Network size	1st decile	1st quartile	Median	3rd quartile	9th decile	Mean
2	57	64	78	98	98	78
3	78	83	90	99	100	90
5	86	88	94	100	100	94
8	85	89	96	100	100	89

Table 4.12: AODV positive response statistics

AODV positive response increased as the network size increased. Each node periodically checked its data request buffer to determine which data requests had not been answered. The protocol re-broadcast the data requests that were not answered. This periodic check ensured that data packets were forwarded, which increased the chance of a positive response to a request.

4.3.4 DSR

4.3.4.1 Total Traffic

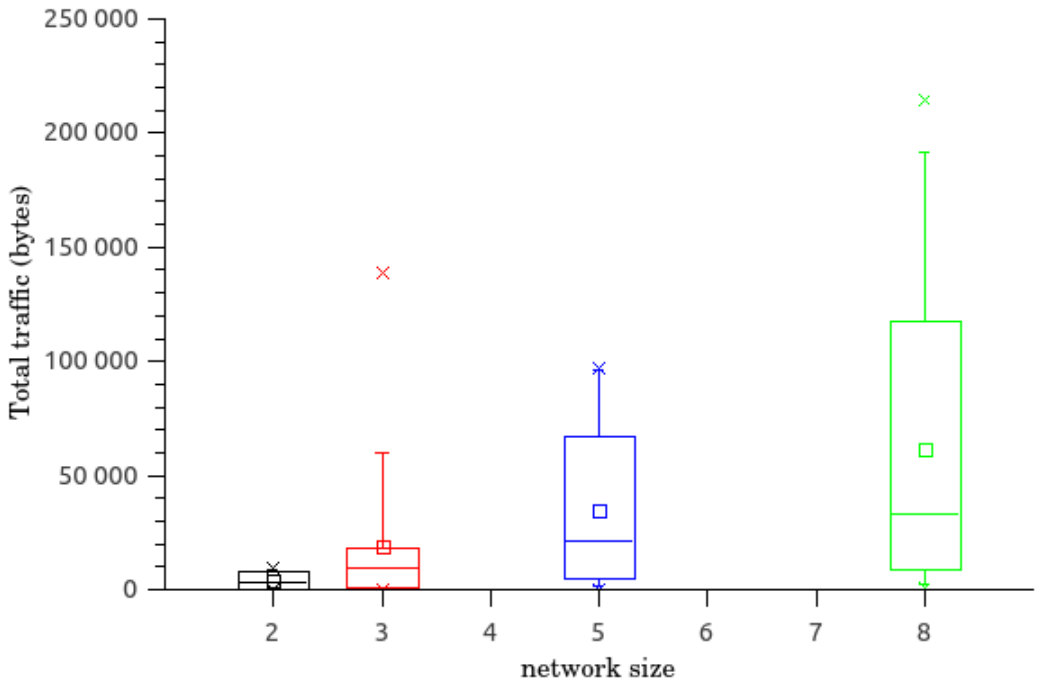


Figure 4.14: DSR total traffic

Network size	1st decile	1st quartile	Median	3rd quartile	9th decile	Mean
2	423,60	1 011,25	2 868,00	7 821,00	8 193,20	4 088,60
3	1 342,10	1 551,50	9 842,00	18 148,00	40 492,00	19 236,60
5	1 447,60	5 152,00	21 612,00	66 728,00	95 857,60	34 991,30
8	4 786,90	9 688,00	33 501,00	117 592,00	129 214,00	61 704,95

Table 4.13: DSR total traffic statistics

Total traffic increased as the network size increased, because as the number of nodes increased in the network, more nodes were able to send packets. The difference in the interquartile range shows that the total traffic increased significantly as the network size increased The zero outliers occurred because nodes moved out of transmission range or were switched off.

4.3.4.2 Data Traffic

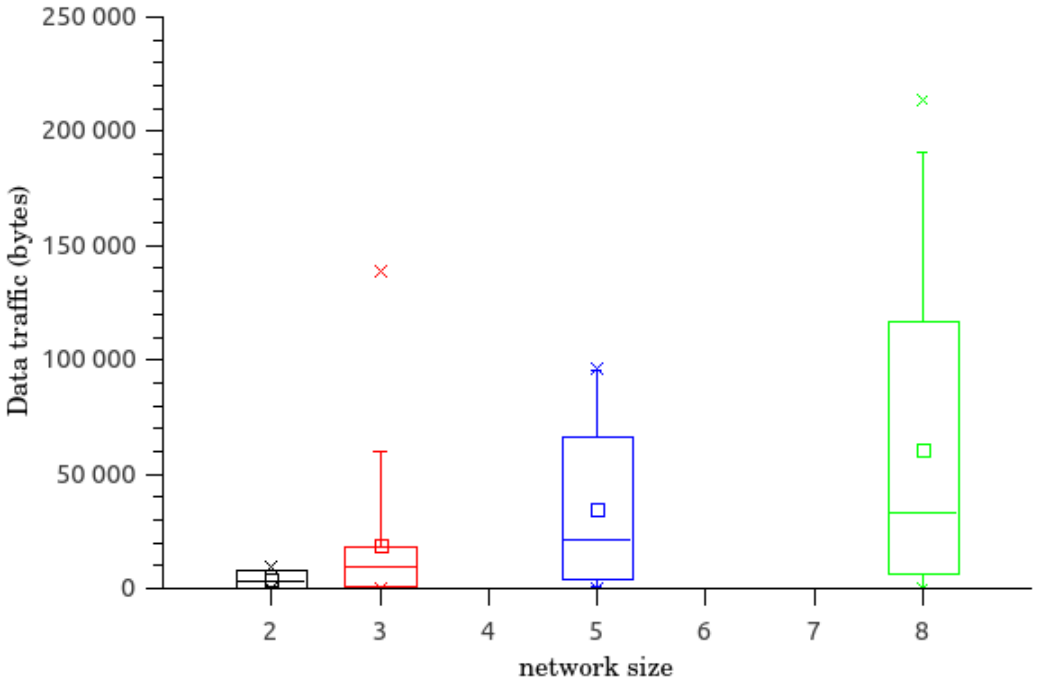


Figure 4.15: DSR data traffic

Network size	1st decile	1st quartile	Median	3rd quartile	9th decile	Mean
2	335,60	923,25	2 780,00	7 733,00	8 105,20	4 001,80
3	1 161,30	1 375,50	9 666,00	17 972,00	40 316,00	19 064,80
5	1 095,60	4 822,00	21 260,00	66 376,00	95 505,60	34 588,10
8	0,00	7 398,00	32 905,00	116 979,00	128 558,80	60 381,15

Table 4.14: DSR data traffic statistics

DSR data traffic behaved similarly to DSR total traffic. As expected, data traffic increased steadily as the network size increased. The zero outliers represent nodes that moved out of transmission range or were switched off.

4.3.4.3 Control Traffic

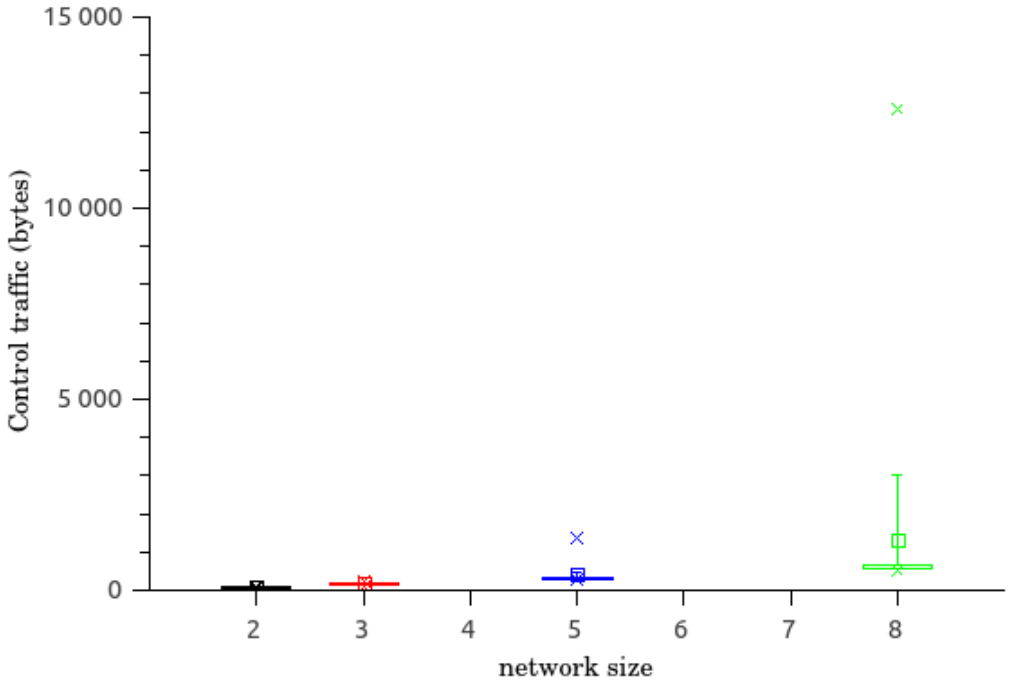


Figure 4.16: DSR control traffic

Network size	1st decile	1st quartile	Median	3rd quartile	9th decile	Mean
2	88,00	88,00	88,00	88,00	88,00	86,80
3	171,60	176,00	176,00	176,00	176,00	171,80
5	352,00	352,00	352,00	352,00	382,40	403,20
8	576,00	611,00	640,00	674,00	918,70	1 323,80

Table 4.15: DSR control traffic statistics

In Figure 4.16, DSR control traffic increased gradually as the network size increased, because route discovery control packets were only sent out when data requests were generated.

4.3.4.4 Delay

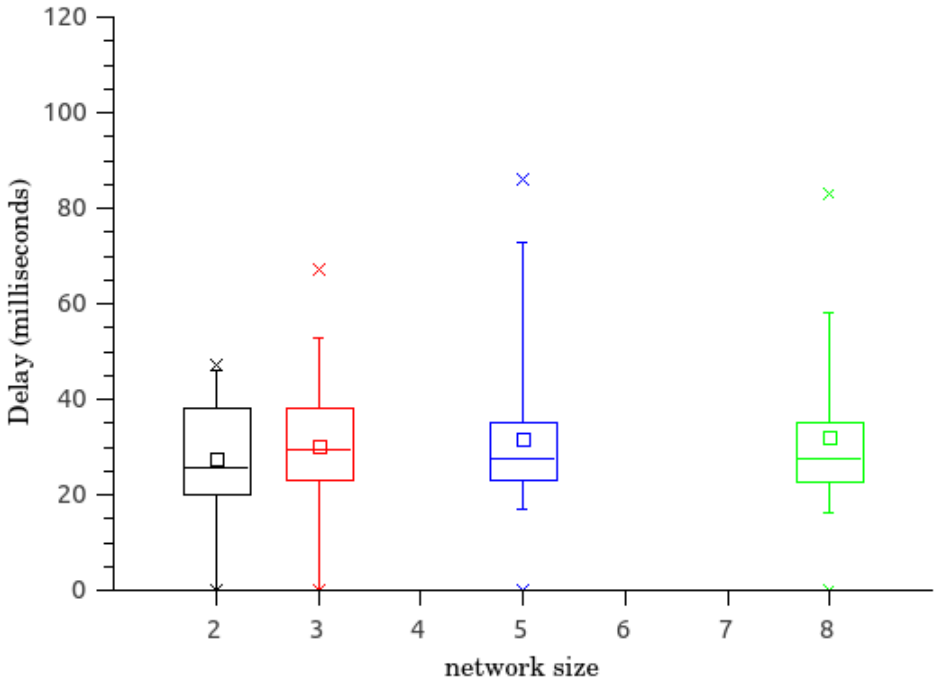


Figure 4.17: DSR delay

Network size	1st decile	1st quartile	Median	3rd quartile	9th decile	Mean
2	13,50	20,25	25,50	38,25	45,10	28
3	16,20	23,25	29,50	38	45,70	30
5	18,90	23,50	27,50	35,25	46,80	32
8	21,50	23	27,50	35,25	48	32

Table 4.16: DSR delay statistics

DSR delay increased slightly as the network size increased. This delay was due to transmission delay, not route discovery delay. Transmission delay transmitted the data traffic. This data traffic was shared between more nodes as the network size increased, thus reducing the delay.

4.3.4.5 Convergence Time

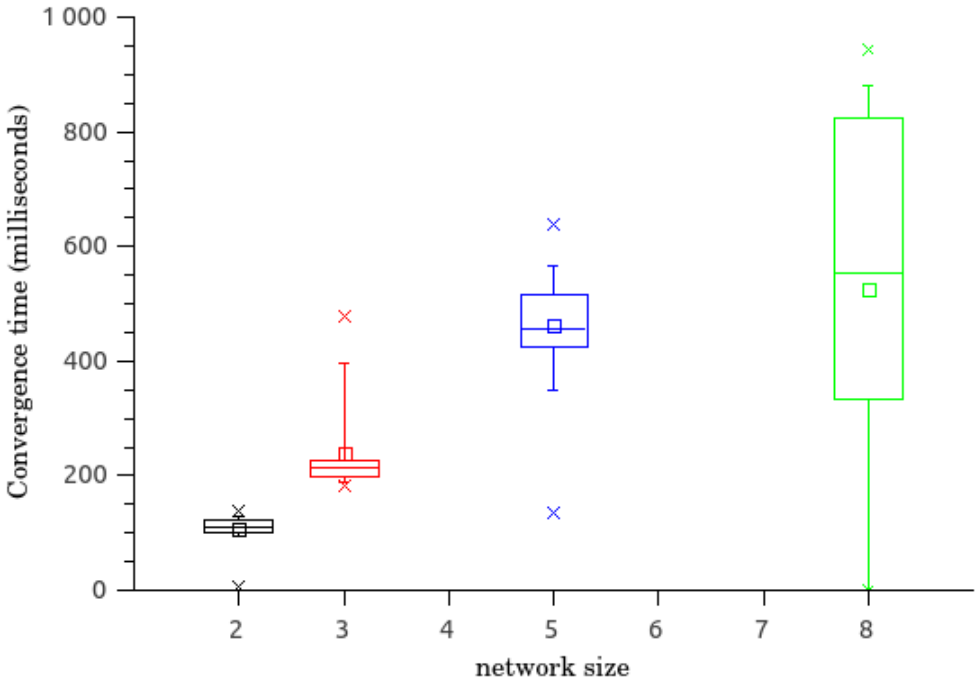


Figure 4.18: DSR convergence time

Table 4.17: DSR convergence time statistics

Network size	1st decile	1st quartile	Median	3rd quartile	9th decile	Mean
2	99,80	103,75	111,50	121,25	125,30	108
3	192,70	202	212,50	225,50	337	238
5	416	428	456,50	514,25	562	462
8	90,90	336,50	553	822,50	876,20	527

DSR convergence time increased greatly as network size increased, because as more nodes were added to the network, more time was needed to discover new destinations and update the routing tables.

4.3.4.6 Positive Response

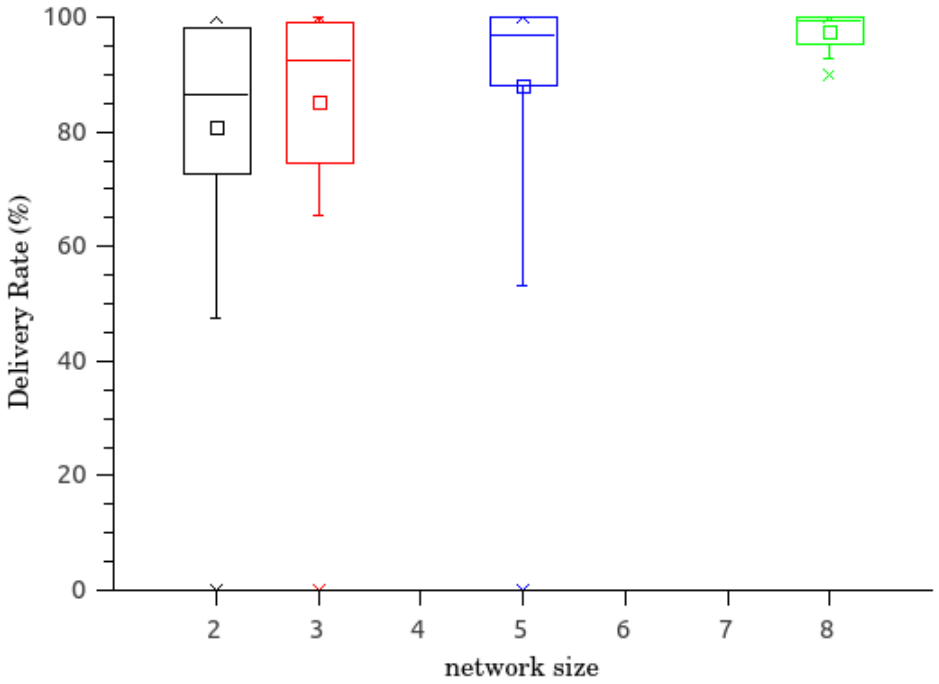


Figure 4.19: DSR positive response

Network size	1st decile	1st quartile	Median	3rd quartile	9th decile	Mean
2	65,30	73	86,50	98	98	81
3	74,40	75	92,50	99	100	85
5	77,60	88,50	97	100	100	88
8	93,90	95,75	99,50	100	100	98

Table 4.18: DSR positive response statistics

The positive response increased as the network size increased. Each node periodically checked its data request buffer to determine which data requests had not been answered. The protocol re-broadcast the data requests that were not answered. This periodic check ensured that data packets were forwarded, which increased the chance of a positive response to a request.

4.3.5 Content-based Routing

4.3.5.1 Total Traffic

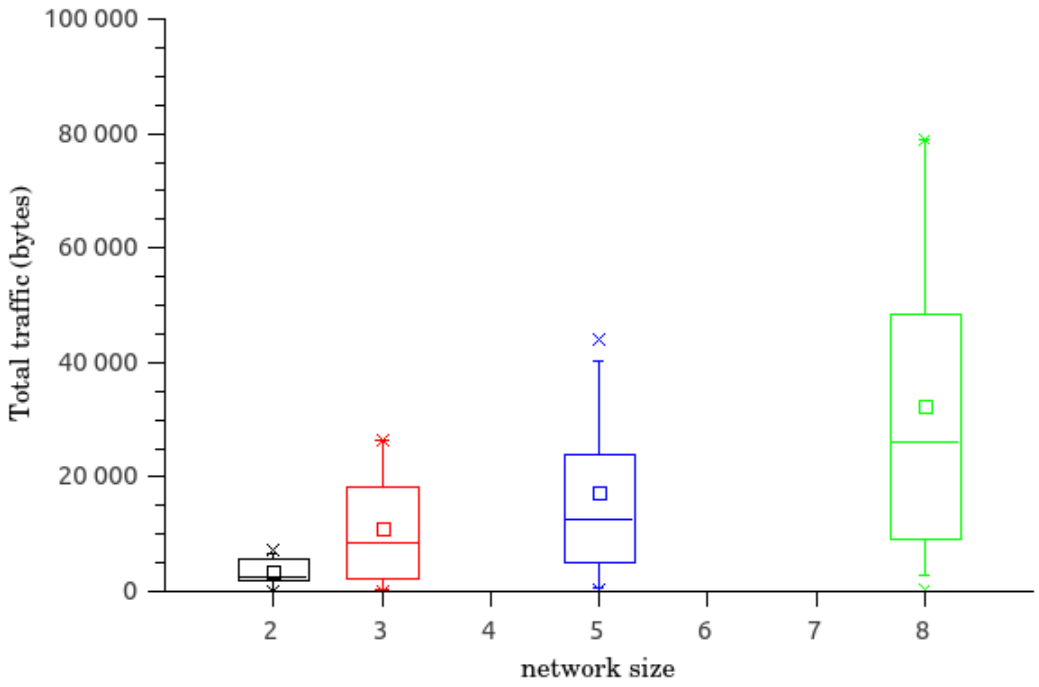


Figure 4.20: Content-based Routing

Network size	1st decile	1st quartile	Median	3rd quartile	9th decile	Mean
2	1 201,20	2 068,75	2 638,00	5 504,00	6 578,40	3 563,30
3	1 616,90	2 440,25	8 455,50	18 104,75	25 777,50	10 924,30
5	3 022,80	5 464,50	12 727,00	23 850,00	39 279,20	17 407,50
8	5 717,40	9 451,00	26 198,50	48 397,50	69 350,40	32 449,55

Table 4.19: CBR total traffic statistics

The number of data packets transmitted increased as the network size increased: more nodes were able to transmit more packets. The zero outliers represented nodes that moved out of transmission range or were switched off.

4.3.5.2 Data Traffic

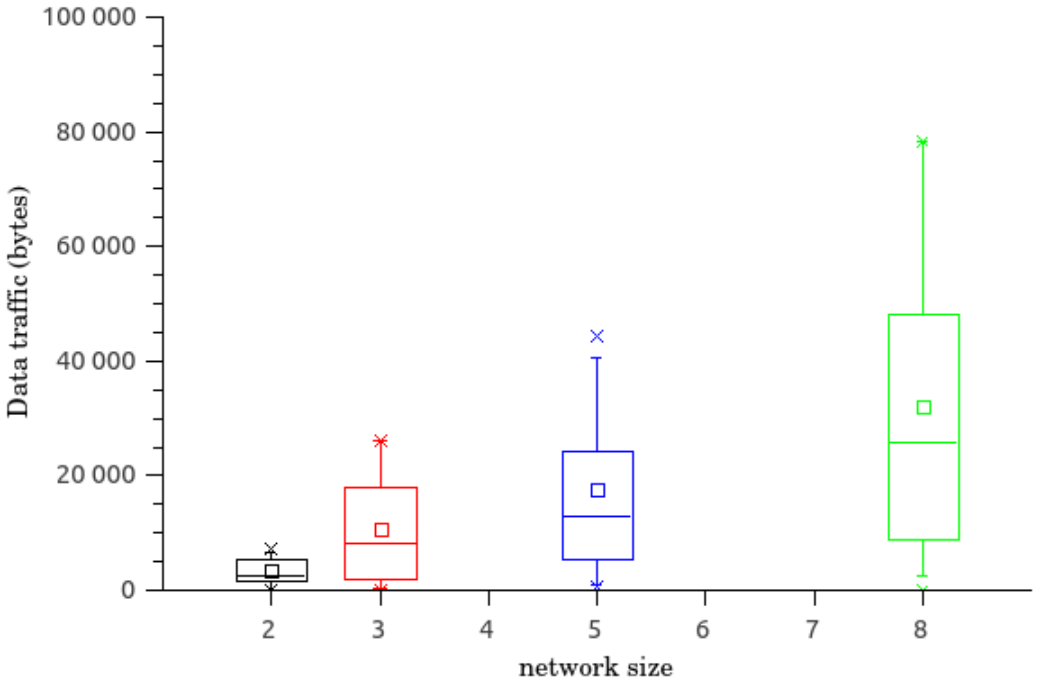


Figure 4.21: Content-based Routing

Network size	1st decile	1st quartile	Median	3rd quartile	9th decile	Mean
2	1 139,20	2 006,75	2 576,00	5 442,00	6 516,40	3 501,30
3	1 492,90	2 316,25	8 331,50	17 980,75	25 653,50	10 800,30
5	3 270,80	5 712,50	12 975,00	24 098,00	39 527,20	17 655,30
8	5 283,40	9 017,00	25 764,50	47 963,50	68 916,40	32 015,55

Table 4.20: CBR data traffic statistics

Content-based routing data traffic increased greatly as the network size increased, because the network exchanged many publication and subscription requests. As calculated from the interquartile range differences, data traffic increased significantly as network size increased. The zero outliers represented nodes that were out of transmission range or were switched off.

4.3.5.3 Control Traffic

Network size	Control traffic (bytes)
2	62
3	124
5	248
8	434

Table 4.21: Content-based routing control traffic

Content-based routing behaved similarly to broadcasting with control traffic. Control traffic increased as the network size increased. A fixed number of control packets were sent out at the beginning of the route discovery process: Content-based routing does not do periodic update messages like AODV, nor initiates route discovery with every data request like DSR.

4.3.5.4 Delay

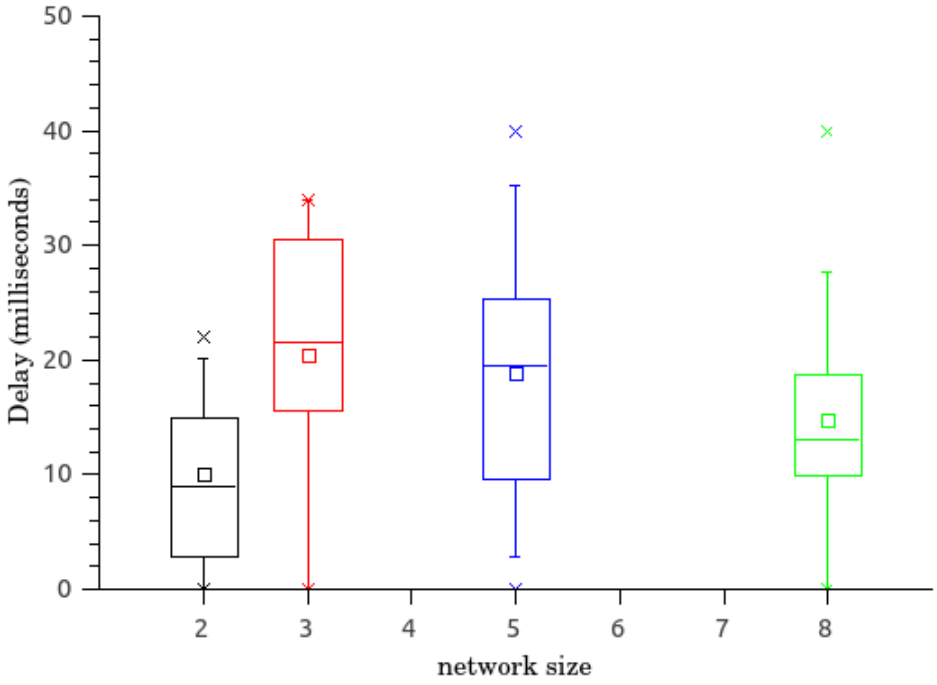


Figure 4.22: Content-based routing delay

Delay increased until network size 3 and then started to decrease. This happened because content-based routing used brokers one hop away to reply to data requests from surrounding nodes. Broker nodes are responsible for storing information published by publishing nodes. A node is always one

Network size	1st decile	1st quartile	Median	3rd quartile	9th decile	Mean
2	0	3	9	15	20	10
3	0	15,75	30,50	21,50	30,50	20
5	4,80	9,75	19,50	25,25	30	19
8	0,90	10	13	18,75	27	15

Table 4.22: Content-based delay statistics

hop away from its broker neighbour and therefore incurs little delay when sending and receiving packets.

4.3.5.5 Convergence Time

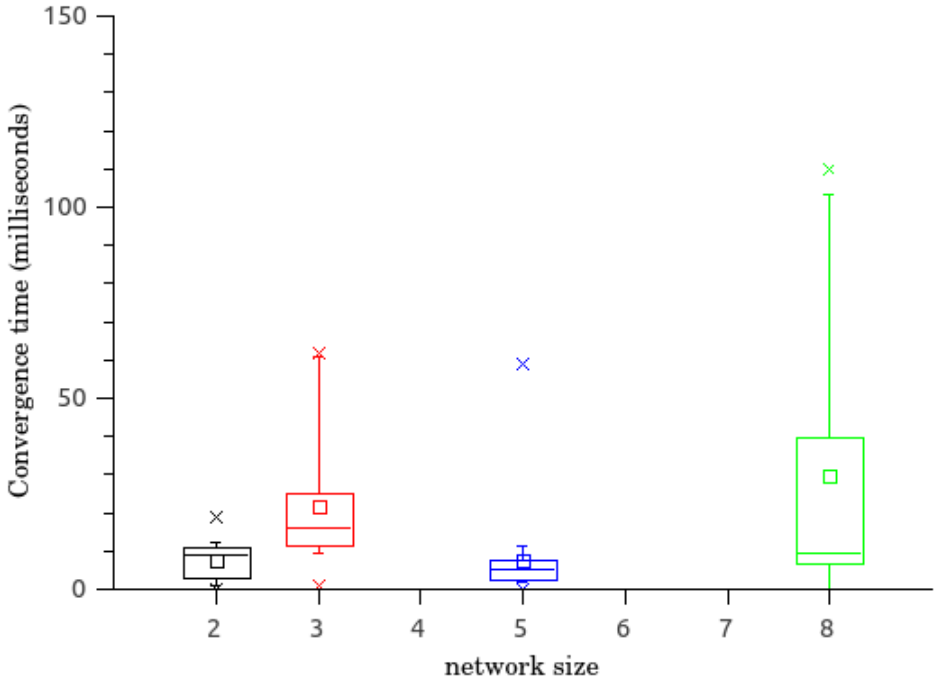


Figure 4.23: Content-based convergence time

Convergence time increased as the network size increased, but very little delay was incurred because publishing and subscribing nodes were only one hop away from the broker nodes, which are responsible for storing all data packets and responding to requests. No route discovery is necessary as nodes are only aware of their one-hop neighbours.

Table 4.23: Content-based convergence time statistics

Network size	1st decile	1st quartile	Median	3rd quartile	9th decile	Mean
2	1	3,25	9	11	11,10	8
3	10	11,75	16	25	43	22
5	2	2,75	5	7,50	9	7
8	5,40	7	9,50	39,75	96,70	30

4.3.5.6 Positive Response

Network size	Positive Response (%)
2	0,24
3	23,02
5	23,96
8	34,41

Table 4.24: Content-based routing positive response

Content-based routing had poor positive response because there was no follow-up procedure to resend data requests that were simply lost with no replies generated.

4.4 Comparison

The previous section presented the results for the different routing protocols in isolation. This section compares the routing protocols according to the performance metrics that were defined. In this section, the results are presented as follows: for each metric defined, box plots comparing the different routing protocols are presented for network size 8. A table of statistics containing the median, mean, interquartile and decile range is also presented. Comparisons are made with reference to the box plots and table of statistics. A bar graph is also presented for each routing metric comparing the different routing protocols.

4.4.1 Total Traffic

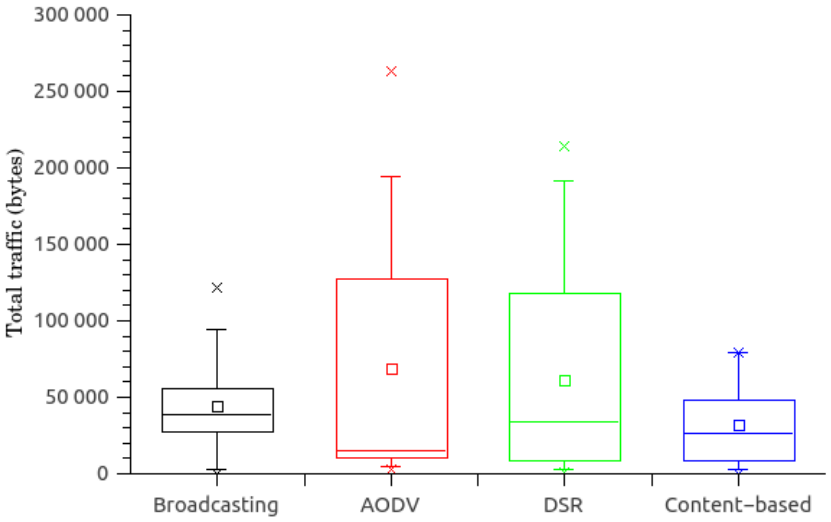


Figure 4.24: Total traffic

Routing protocol	1st decile	1st quartile	Median	3rd quartile	9th decile	Mean
Broadcasting	7 765,90	28 638,75	38 808,00	55 606,75	89 243,70	44 073,25
AODV	6 002,40	11 620,50	15 146,00	127 756,50	189 249,80	69 038,35
DSR	4 786,90	9 688,00	33 501,00	117 592,00	129 214,00	61 704,95
Content-based	5 717,40	9 451,00	26 198,50	48 397,50	69 350,40	32 449,55

Table 4.25: Total traffic, network size 8 statistics

AODV total traffic was highest, as indicated by the mean. This was because AODV generated more control traffic than the other routing protocols. DSR generated the second highest total traffic because it generated control traffic every time a new packet needed to be sent to an unknown destination. Broadcasting and content-based routing generated the least amount of total traffic because these protocols did not initiate route discovery every time a new packeted needed to be delivered.

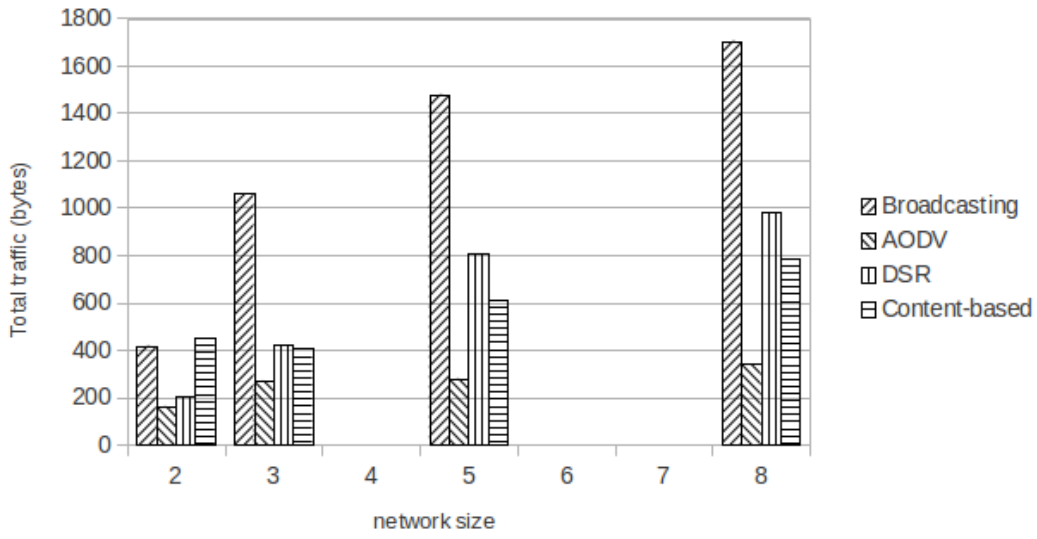


Figure 4.25: Total traffic vs network size

As seen in Figure 4.25, AODV and DSR showed significant total traffic across the different network sizes because they generated more control traffic. Content-based routing total traffic increased gradually as the network size increased. Nodes assigned the broker or server role were responsible for responding to data requests and not forwarding requests unless they did not have the data requested. Broadcasting had slightly more total traffic than content-based routing because it forwarded a request to all connected neighbours instead of choosing which node to forward the request or packet to.

4.4.2 Data Traffic

AODV and DSR had roughly the highest and same amount of data traffic for network size 8. More connections per node meant AODV and DSR were able to transmit a proportional amount of packets. Broadcasting data traffic was slightly less than AODV and DSR but significantly more than content-based routing. Data traffic in content-based routing only travelled one hop to the closest broker node. As a result less data were transmitted through the network.

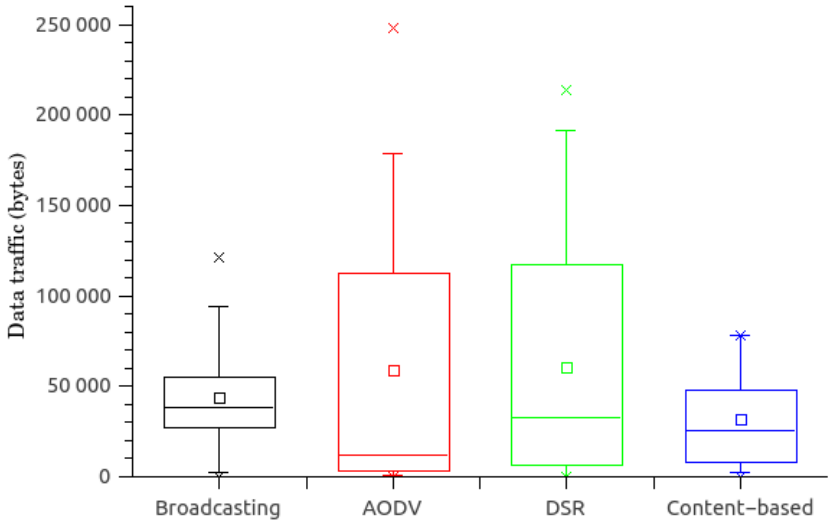


Figure 4.26: Data traffic

Routing protocol	1st decile	1st quartile	Median	3rd quartile	9th decile	Mean
Broadcasting	7 331,90	28 204,75	38 374,00	55 172,75	88 809,70	43 639,25
AODV	3 088,90	4 344,75	11 955,00	112 553,00	161 246,80	60 381,15
DSR	0,00	7 398,00	32 905,00	116 979,00	128 558,80	58 926,85
Content-based	5 283,40	9 017,00	25 764,50	47 963,50	68 916,40	32 015,55

Table 4.26: Data traffic, network size 8 statistics

Figure 4.27 shows that as the network size increased, AODV, DSR and broadcasting data traffic increased significantly. Content-based routing only transmitted data to the closest one-hop neighbour and did not propagate a packet like broadcasting throughout the network, or like AODV and DSR that forward a packet until its TTL has expired or it has reached its intended destination.

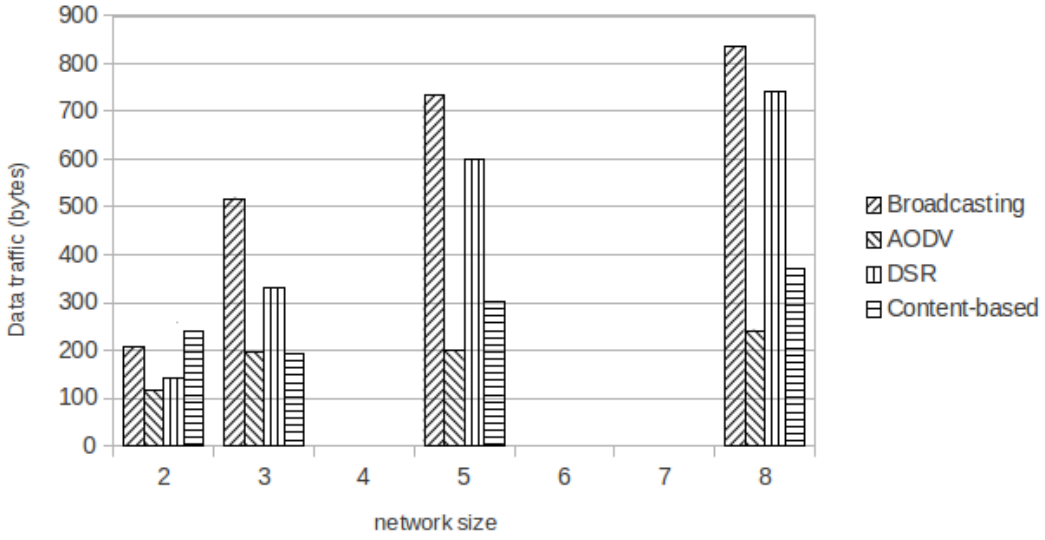


Figure 4.27: Data traffic vs. number of nodes

4.4.3 Control Traffic

Network size	Broadcasting	AODV	DSR	Content-based
2	1	2	2	1
3	1.3	2.3	2.3	1.3
5	1.6	2.6	2.6	1.6
8	1.625	2.625	2.625	1.625

Table 4.27: Average routing table size

In this section, routing table sizes are also shown in Table 4.27 to explain the behaviour of control traffic in the different routing protocols. The control traffic incurred by AODV is significant, as shown in Figure 4.28. AODV's high control traffic is accompanied by a large routing table length. AODV has more control traffic because of the route discovery and route maintenance processes initiated periodically. DSR table length is similar to AODV because DSR initiated a route discovery every time a new data request was generated, but DSR control traffic remained significantly less than AODV because control packets were not generated periodically. Control traffic for broadcasting and content-based routing was significantly less than for AODV. Broadcasting and content-based routing only initiated route discovery at the very beginning of the network and never again. No route

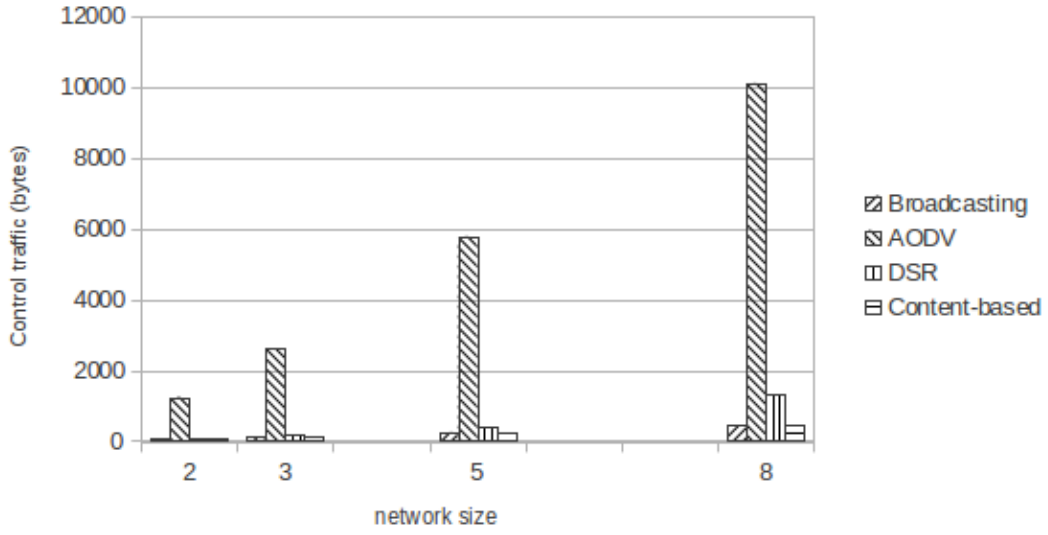


Figure 4.28: Control traffic

maintenance was initiated either. Their table lengths indicate that only connected neighbouring information was stored in the network. Their routing table lengths consequently satisfied the upper limit of $\Theta\sqrt{nlogn}$.

4.4.4 Delay

AODV and DSR showed greater delay. This was due to routing decisions needed to be made by consulting the routing tables. The AODV routing table may sometimes contain outdated routes or no desired route. Thus, the route discovery procedure and route maintenance would consume more time, resulting in greater delay when compared to other routing protocols. Broadcasting delay is lowest. No local node routing decisions are needed to forward a message. This delay is simply attributed to the time it takes for the message to travel along the wireless channel. DSR delay is second highest. This can also be attributed to routing decisions that have to be made when a node needs to forward a message. The node needs to check its local routing table to determine if a route to the destination specified in the message still exists.

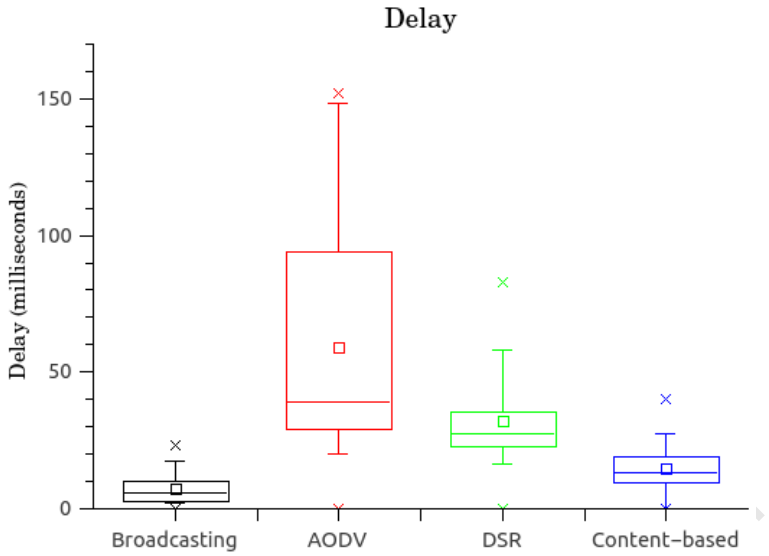


Figure 4.29: Delay

Routing protocol	1st decile	1st quartile	Median	3rd quartile	9th decile	Mean
Broadcasting	2	3	6	10	11,60	7
AODV	25	29,75	39	94	110,20	59
DSR	21,50	23	27,50	35,25	48	32
Content-based	0,90	10	13	18,75	27	15

Table 4.28: Delay, network size 8 statistics

Previous literature describes that in stable networks, the arrival rate of messages is equal to the rate of departure (if unequal an unstable network is observed). Using Little's theorem [5, 24, 26, 66] commonly used in queuing theory, the average number of messages in a network of size n at any node is equal to the average rate of arrival, or in the case of the traffic indicated above, (T_T) over time t. Delay has an upper bound inversely proportional to the throughput upper bound, that is, $\Theta(\sqrt{nlogn})$, were n is the number of nodes. Figure 4.32 shows that in AODV and DSR, this relationship is proven. Broadcasting and content-based routing on the other hand, show a decrease in delay as network size increases. Shorter delays occur as more nodes are available to relay messages quickly.

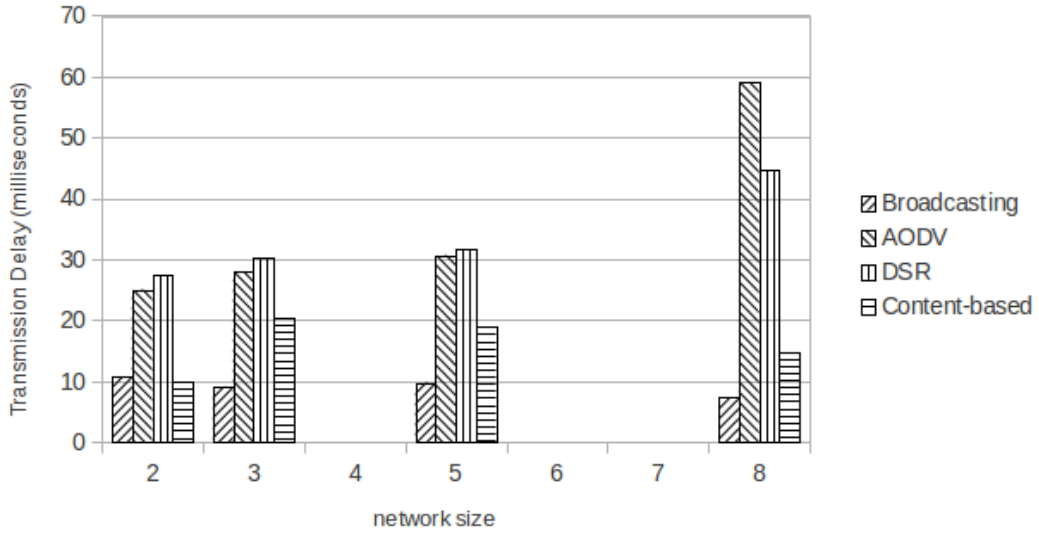


Figure 4.30: Delay vs. network size

4.4.5 Convergence Time

Routing Protocol	1st decile	1st quartile	Median	3rd quartile	9th decile	Mean
Broadcasting	1,80	10	19	46	55,70	29
AODV	105,80	506,25	713	861,25	937	631
DSR	90,90	336,50	553	822,50	876,20	527
Content-based	5,90	7	9,50	39,75	96,70	30

Table 4.29: Convergence time, network size 8 statistics

As Figure 4.32 shows, AODV and DSR had a longer convergence time caused by the route discovery process initiated by every node to which a packet is forwarded to discover the intended destination. Time was used to update routing tables with new routes.

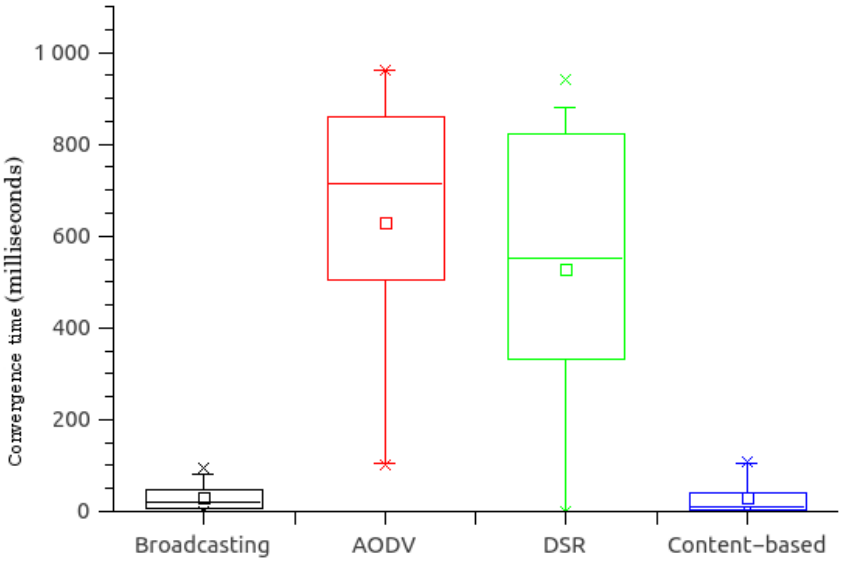


Figure 4.31: Convergence time

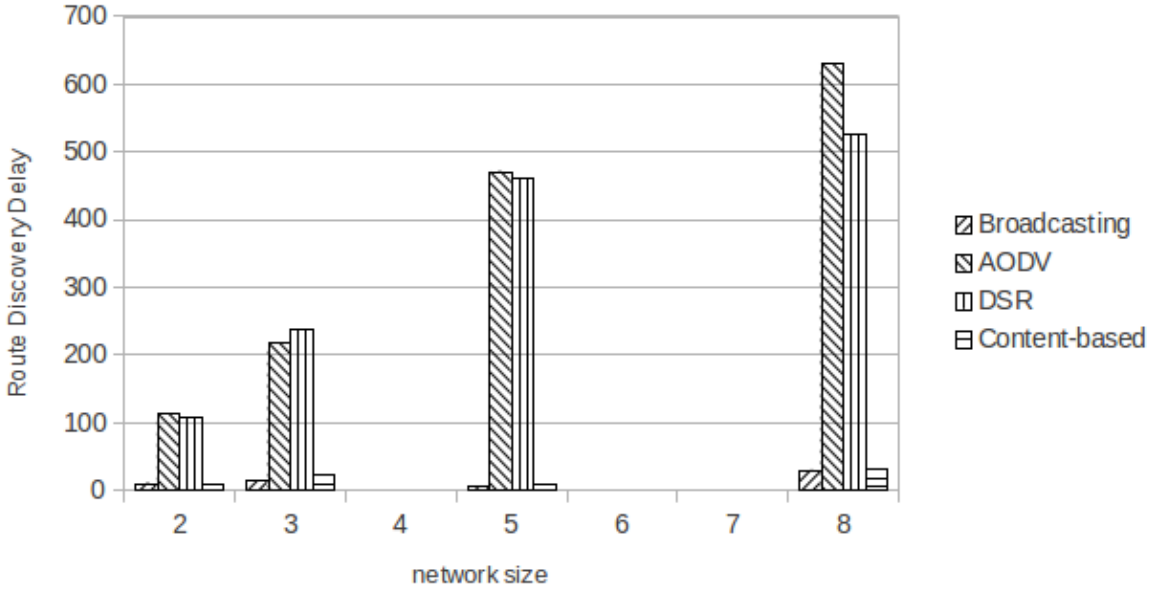


Figure 4.32: Convergence time vs. network size

4.4.6 Positive Response

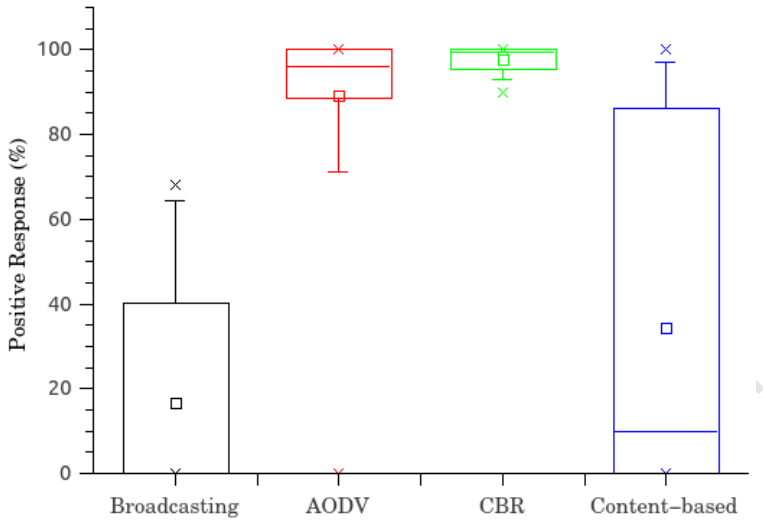


Figure 4.33: Positive response

Routing protocol	1st decile	1st quartile	Median	3rd quartile	9th decile	Mean
Broadcasting	0,00	0,00	0,00	40,25	60,40	16,25
AODV	84	89	96	100	100	89
DSR	93,90	95,75	99,50	100	100	98
Content-based	0,00	0,00	10,00	86,25	97,00	34,41

Table 4.30: Positive response, network size 8 statistics

AODV and DSR a had better positive response. This is because these protocols periodically checked their route and data request buffers during the route discovery and data requesting process, as well as recent packets not answered, increasing their chances of receiving a response about the data.

Network size	Broadcasting	AODV	DSR	Content-based
2	9,42	78	81	0,24
3	10,42	90	85	23,02
5	26,12	94	88	23,96
8	16,52	89	98	34,41

Table 4.31: Overall positive response (%)

Table 4.31 shows the observed positive response as the network size increased. Broadcasting and content-based routing had the lowest positive response. The positive response for AODV and DSR was greatest.

4.5 Discussion

This chapter described the experiment setup and presented the results obtained after evaluation of the performance of routing metrics defined. Table 4.32 summarises the results that were obtained during emulation for each protocol at network size 8. A discussion of the results follows.

	Broadcasting	AODV	DSR	Content-based
Total Traffic (bytes)	44 073,25	69 038,35	61 704,95	32 449,55
Data Traffic (bytes)	43 639,25	60 381,15	58 926,85	32 015,55
Control Traffic (bytes)	434	10 111,50	1 323,80	434
Delay (milliseconds)	7	59	32	15
Convergence Time (milliseconds)	29	631	527	30
Positive Response (%)	16,52,	89	98	34,41
Routing Table Length	1,6	2,6	2,6	1,6

Table 4.32: Overall results

The results show that AODV and DSR have the highest total traffic. These protocols also have a significant amount of control traffic. This control traffic is due to the control packets generated during route discovery and route maintenance.

The total traffic in broadcasting and content-based routing is consumed by data traffic. These protocols only generate a fix number of control packets at the start of the communication session between nodes. Afterwards, no route maintenance or route discovery is initiated. Therefore, control traffic makes up a small proportion of the total traffic.

The low control traffic figures for broadcasting and content-based routing are associated with a small number of entries in the routing tables. AODV and DSR have a high number of route entries in each node's routing table. The number of entries in the AODV and DSR routing table can grow up to the size of the network.

AODV and DSR incurred the highest delay and convergence times. AODV and DSR delay was influenced by the routing decisions that were made at the nodes. The convergence time for AODV and DSR is high because the route discovery process, which is initiated first, takes longer to discover new routes and update routing tables.

Broadcasting and content-based routing perform roughly the same on account of delay and convergence time. Delay is less than when using AODV and DSR because nodes do not deliberate about where to send a message or consult routing tables.

Broadcasting and content-based routing had the lowest convergence times, meaning these proto-

cols establish a network topology fairly quickly. In addition, these protocols do not initiate route discovery like AODV and DSR. AODV and DSR have high convergence times, taking longer to connect nodes in the network.

The high positive response observed for AODV and DSR is due to the rebroadcasting of data requests that remained unanswered in the data request buffer. Broadcasting and content-based routing only send out a data request once and do not make repeated attempts to resend the request if no response is received.

Chapter 5

Conclusion

This research considered the development of a mobile phone Bluetooth ad hoc network to be used in opportunistic communication between mobile phones in the same vicinity. The Bluetooth ad hoc network allows mobile phones to share information in a P2P manner. Participating mobile phones take into consideration the master/slave relationship imposed by the Bluetooth protocol. The research therefore first considered the feasibility of creating a multi-hop ad hoc network with this Bluetooth master/slave relationship. The research further considered how to route information in the Bluetooth ad hoc network efficiently. Four representative routing protocols from the P2P and mobile ad hoc networking paradigms were chosen for implementation and evaluated to determine how efficiently they route information in the Bluetooth ad hoc network.

Previous research presented many routing solutions for wireless mobile ad hoc networks but through theoretical review it was discovered that not many routing solutions exist for opportunistic mobile ad hoc networks such as those consisting only of mobile phones. In Chapter 2, the strengths and weaknesses of existing routing protocols in mobile ad hoc networks and lookup or routing solutions in the P2P paradigm are presented. From the literature review, four routing protocols were chosen for evaluation: AODV, DSR, broadcasting and finally a content-based routing protocol.

In order to be able to study the feasibility of creating a multi-hop Bluetooth ad hoc network that uses different routing protocols, a prototype mobile application was designed. This mobile application is easily deployed on mobile phones and allows mobile phones to form an ad hoc network using Bluetooth's node discovery procedure, also known as device inquiry. Chapter 3 presented the design of the multi-hop topology envisioned for the Bluetooth ad hoc network. The design of the mobile application and its layering in the TCP/IP protocol stack were presented. The user interface functions and routing functionality were presented in this chapter, as well as their positions on the protocol stack.

In chapter 4, the implementation of the prototype mobile application was presented. The routing functionality, user interface and data lookup functions were shown in this chapter. In Chapter 4, the feasibility study of the first prototype implementation discussed the challenged experienced during testing on a test bed of Nokia N96 mobile phones.

The final aim of this research was to study the performance of the various routing protocols for

Bluetooth multi-hop ad hoc networks using various topologies. Unlike the feasibility study, experimentation in a controlled environment obtained less varied results, which were used to compare the performance of the routing protocols according to the total traffic, data traffic, delay, positive response, control traffic and routing table size.

The research had its limitations however: because a small, stable network was assumed, dynamic node behaviour was modelled by switching off devices to simulate nodes moving out of transmission range or users switching off their devices. Experiments were conducted using emulation, as test bed results were influenced by factors such as human intervention and the heavy burden of running a multi-threaded application on mobile phones with limited processing capacity. These challenges made test bed testing difficult to monitor. However, the prototype mobile application developed using the J2ME platform was easy to deploy on mobile phones to create an ad hoc network of mobile phones. File sharing was possible in this Bluetooth ad hoc network and on the test bed.

The evaluation of the performance routing metrics is used to answer the research questions initially posed. From the results, the research discusses the findings, challenges and limitations experienced in an attempt to answer the research questions.

1. *What is the feasibility of peer-to-peer file sharing in a Bluetooth ad hoc network?*

   It is possible to implement P2P roles between mobile phones in a Bluetooth ad hoc network. With the modular design of the system, nodes communicate in a P2P manner at the application layer, implementing route discovery, route maintenance, advertising device local address for other nodes to discover, send and receive data in a P2P manner. However, at the data link layer, Bluetooth imposes a master/slave relationship between devices. In order to form a multi-hop ad hoc network, the master/slave relationship was used to form interconnected piconets called scatternets. The feasibility study highlighted that implementing the functionality of route discovery and advertising of a local address using multi-threading is not an optimal solution. The design overcame this problem by using a client/server approach at the data link layer. This approach fit in perfectly with the Bluetooth master/slave feature. The master device, with server role, became responsible for advertising its local address. The slave device was responsible for route discovery and the rest of the functions in the client role. All this was hidden from the user by showing the master and slave devices sharing the same functions and communicating in P2P manner.

2. *How do existing routing protocols perform in the multi-hop Bluetooth ad hoc network?*

   AODV and DSR had the highest total traffic but it consisted mainly of control traffic. AODV and DSR were also the worst performing in terms of delay and convergence times. AODV and DSR had the highest control traffic due to route discovery processes initiated when a destination was unknown. AODV and DSR also had the greatest routing table length compared to broadcasting and content-based routing. Broadcasting and content-based routing

are only concerned with delivering packets to one-hop neighbours and maintain very little routing information in their routing tables. These protocols also had the shortest delay and convergence times. However, broadcasting and content-based routing had the worst positive response because these protocols made little attempt at trying to receive responses from destinations to which data requests had been sent. These observations suggest that broadcasting and content-based routing seem more suitable for information dissemination in sparse, stable mobile phone Bluetooth ad hoc networks.

In conclusion, in order to determine which routing protocols are suitable for real world settings, the routing protocols need to be tested in a testbed consisting of multiple mobile devices. A future study would implement a mobility model in the real world to see how these routing protocols respond to disruptions in the network.

References

[1] Smart, The Next Wave of Bluetooth. `http://www.abiresearch.com/research/product/1013429-smart-the-next-wave-of-bluetooth/`. Accessed: 1 February 2013.

[2] A. Abada, C. Huang, L. Cui, and H. Chen. A Novel Path Selection and Recovery Mechanism for MANETs P2P File Sharing Applications. *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC'07)*, 1:3475–3480, 2007.

[3] K. Aberer and M. Hauswirth. An Overview on Peer-to-Peer Information Systems. In *Workshop on Distributed Data and Structures WDAS-2002*, 2002.

[4] M. Abolhasan, T. Wysocki, and E. Dutkiewicz. A Review of Routing Protocols for Mobile Ad Hoc Networks. *Ad Hoc Networks*, 2(1):1–22, 2004.

[5] Gahng-Seop Ahn, A.T. Campbell, A. Veres, and Li-Hsiang Sun. Supporting Service Differentiation for Real-time and Best-effort Traffic in Stateless Wireless Ad Hoc Networks (SWAN). *Mobile Computing, IEEE Transactions on*, 1(3):192 – 207, Jul-Sep 2002.

[6] J. E. Anholt, V. Grigoreanu, T. Likarish, and B. Risteska. Look-Ahead Routing Reduces Wrong Turns in Freenet-Style Peer-to-Peer Systems. *Proceedings of the 38th Hawaii International Conference on System Sciences*, 2005.

[7] Osamah S. Badarneh and Michel Kadoch. Multicast Routing Protocols in Mobile Ad Hoc Networks: A Comparative Survey and Taxonomy. *EURASIP Journal on Wireless Communications and Networking*, 2009(764047), 2009.

[8] Fan Bai, Narayanan Sadagopan, and Ahmed Helmy. The Important Framework for Analyzing the Impact of Mobility on Performance of Routing protocols for Adhoc Networks. *Ad Hoc Networks*, 1(4):383–403, 2003.

[9] R. Baldoni and A. Virgillito. Distributed Event Routing in Publish/Subscribe Communication Systems: A Survey. *DIS, Universita di Roma La Sapienza, Technical Report*, 2005.

[10] S. Basagni and C. Petrioli. Multihop Scatternet Formation for Bluetooth Networks. In *Vehicular Technology Conference, 2002. VTC Spring 2002. IEEE 55th*, volume 1, pages 424–428, 2002.

[11] J. E. Berkes. Decentralized peer-to-peer network architecture: Gnutella and freenet. *University of Monitoba Winnipeg, Manitoba, Canada*, 2003.

[12] E. Bruns, B. Brombach, T. Zeidler, and O. Bimber. Enabling Mobile Phones To Support Large-Scale Museum Guidance. *MultiMedia, IEEE*, 14(2):16–25, April-June 2007.

[13] Antonio Carzaniga, David S. Rosenblum, and Alexander L. Wolf. Design and Evaluation of a Wide-Area Event Notification Service. *ACM Transactions on Computer Systems (TOCS)*, 19(3):332–383, Aug 2001.

[14] Yatin Chawathe, Sylvia Ratnasamy, Lee Breslau, Nick Lanham, and Scott Shenker. Making Gnutella-like P2P Systems Scalable. In *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '03, pages 407–418, New York, NY, USA, 2003. ACM.

[15] I. Chlamtac, M. Conti, and J. J. Liu. Mobile Ad Hoc Networking: Imperatives and Challenges. *Ad Hoc Networks*, 1(7):13–64, 2003.

[16] I. Clarke, S.G. Miller, T.W. Hong, O. Sandberg, and B. Wiley. Protecting free expression online with Freenet. *Internet Computing, IEEE*, 6(1):40–49, Jan/Feb 2002.

[17] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. Freenet: A Distributed Anonymous Information Storage and Retrieval System. *Proceedings of the ICSI Workshop on Design Issues in Anonymity and Unobservability*, 2000.

[18] Ian Clarke. Freenet's Next Generation Routing Protocol. `https://freenetproject.org/ngrouting.html`. Accessed: 1 February 2013.

[19] M.S. Corson, J.P. Macker, and G.H. Cirincione. Internet-based Mobile Ad Hoc Networking. *Internet Computing, IEEE*, 3(4):63–70, Jul–Aug 1999.

[20] F. Cuomo, G. Di Bacco, and T. Melodia. SHAPER: A Self-Healing Algorithm Producing Multi-hop Bluetooth Scatternets. In *Global Telecommunications Conference, GLOBECOM '03. IEEE*, volume 1, pages 236–240, Dec. 2003.

[21] Gang Ding and Bharat Bhargava. Peer-to-Peer File-sharing over Mobile Ad Hoc Networks. *Proceedings of the 2nd IEEE Conference on Pervasive Computing and Communications*, 2004.

[22] Davide Frey and Gruia-Catalin Roman. Context-aware Publish Subscribe in Mobile Ad Hoc Networks. In *Proceedings of the 9th International Conference on Coordination Models and Languages*, COORDINATION'07, pages 37–55, Berlin, Heidelberg, 2007. Springer-Verlag.

[23] M. Frodigh, P. Johansson, and P. Larsson. Wireless Ad Hoc Networking – The Art of Networking without a Network. *Ericsson Review*, 4(4):249, 2000.

[24] A.El. Gamal, J. Mammen, B. Prabhakar, and D. Shah. Throughput-Delay Trade-Off in Wireless Networks. In *INFOCOM 2004. Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 1, pages –475, March 2004.

[25] S. Ghandeharizadeh and Bhaskar Krishnamachari. C2P2: A Peer-to-Peer Network for On-Demand Automobile Information Services. In *Database and Expert Systems Applications, 2004. Proceedings. 15th International Workshop on*, pages 538–542, Aug-Sept 2004.

[26] Robin Groenevelt, Philippe Nain, and Ger Koole. The Message Delay in Mobile Ad Hoc Networks. *Performance Evaluation*, 62(1–4):210–228, Oct 2005.

[27] R.A. Guérin, J. Rank, S. Sarkar, and E. Vergetis. Forming Connected Topologies in Bluetooth Ad-hoc Networks–An Algorithmic Perspective. 5b:1011–1020, 31 August–5 September 2003.

[28] Z.J. Haas. A New Routing Protocol for the Reconfigurable Wireless Networks. In *Universal Personal Communications Record, 1997. Conference Record., 1997 IEEE 6th International Conference on*, volume 2, pages 562–566, October 1997.

[29] J. Hoebeke, I. Moerman, B. Dhoedt, and P. Demeester. An Overview of Mobile Ad Hoc Networks: Applications and Challenges. *Communications Network Journal*, 3(3):60–66, 2004.

[30] T. Hossfeld, K. Tutschku, F.-U. Andersen, H. de Meer, and J.O. Oberender. Simulative Performance Evaluation of a Mobile Peer-to-Peer File-sharing System. In *Next Generation Internet Networks, 2005*, pages 281–287, April 2005.

[31] Yongqiang Huang and Hector Garcia-Molina. Publish /Subscribe in a Mobile Environment. *Wireless Networks*, 10:643–652, 2004. 10.1023/B:WINE.0000044025.64654.65.

[32] Shengming Jiang, Dajiang He, and Jianqiang Rao. A Prediction-based Link Availability Estimation for Mobile Ad Hoc Networks. In *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1745 –1752 vol.3, 2001.

[33] Sewook Jung, Uichin Lee, Alexander Chang, Dae-Ki Cho, and Mario Gerla. BlueTorrent: Cooperative Content Sharing for Bluetooth Users. *Pervasive and Mobile Computing*, 3(6):609–634, 2007.

[34] Daishi Kato, Kaoutar Elkhiyaoui, Kazuo Kunieda, Keiji Yamada, and Pietro Michiardi. A Scalable Interest-Oriented Peer-to-Peer Pub/Sub Network. *Peer-to-Peer Networking and Applications Journal, Springer*, December 2010.

[35] N. Kayastha, D. Niyato, P. Wang, and E. Hossain. Applications, Architectures, and Protocol Design Issues for Mobile Social Networks: A Survey. *Proceedings of the IEEE*, 99(12):2130–2158, Dec 2011.

[36] W. Kellerer, Z. Despotovic, M. Michel, Q. Hofstatter, and S. Zols. Towards a Mobile Peer-to-Peer Service Platform. In *Applications and the Internet Workshops, 2007. SAINT Workshops 2007. International Symposium on*, pages 2–2. IEEE, 2007.

[37] M. Khabbazian and V.K. Bhargava. Efficient Broadcasting in Mobile Ad Hoc Networks. *Mobile Computing, IEEE Transactions on*, 8(2):231–245, Feb. 2009.

[38] A. Khetrapal. Routing Techniques for Mobile Ad Hoc Networks Classification and Qualitative/Quantitative Analysis. *Proceedings of ICWN*, pages 251–257, 2006.

[39] A. Klemm, C. Lindemann, and O. P. Waldhorst. A Special-purpose Peer-to-Peer File Sharing System for Mobile Ad Hoc Networks. *Vehicular Technology Conference, 2003. VTC 2003-Fall. 2003 IEEE 58th*, 2003.

[40] G. Kortuem, J. Schneider, D. Preuitt, T.G.C Thompson, S. Fickas, and Z. Segall. When Peer-to-Peer comes Face-to-Face: Collaborative Peer-to-Peer Computing in Mobile Ad-hoc Networks. *Proceedings of the 1st International Conference on Peer-to-Peer Computing*, pages 75–91, 2001.

[41] Ching Law and Kai-Yeung Siu. A Bluetooth Scatternet Formation Algorithm. In *Global Telecommunications Conference, 2001. GLOBECOM '01. IEEE*, volume 5, pages 2864–2869, 2001.

[42] Hanh Le, Nina Schiff, Johan du Plessis, and Doan Hoang. A Pervasive Tele-health System for Continual and Low Intrusive Monitoring using Peer-to-Peer Networks. *International Journal Computer Applications in Technology*, 34:330–334, 2009.

[43] David Liben-Nowell, Hari Balakrishnan, and David Kruger. Analysis of the Evolution of Peer-to-Peer Systems. *Proceedings of the 21st Annual Symposium on Principles of Distributed Computing*, pages 233–242, 2002.

[44] C. Lindemann and O. P. Waldhorst. A Distributed Search Service for Peer-to-Peer File Sharing in Mobile Applications. *Proceedings of the Second International Conference on Peer-to-Peer Computing*, pages 73–80, 2002.

[45] Eng Keong Lua, Jon Crowcroft, Marcelo Pias, and Ravi Sharma. A Survey and Comparison of Peer-to-Peer Overlay Network Schemes. *IEEE Communications and Serverys*, 7(2):72–93, 2005.

[46] Qin Lv, Pei Cao, Edith Cohen, Kai Li, and Scott Shenker. Search and replication in unstructured peer-to-peer networks. In *Proceedings of the 16th International Conference on Supercomputing*, ICS '02, pages 84–95, New York, NY, USA, 2002. ACM.

[47] W. Mapham. Mobile phones: Changing Health care One SMS at a time. *Southern African Journal of HIV Medicine*, 9(4):11, 2008.

[48] A. Marques, F. Mira da Silva, and R. Rocha. P2P over Mobile Ad-hoc Networks. In *Sensor, Mesh and Ad Hoc Communications and Networks Workshops, 2009. SECON Workshops '09. 6th Annual IEEE Communications Society Conference on*, pages 1–3, June 2009.

[49] P. McDermott-Wells. What is Bluetooth? *Potentials, IEEE*, 23(5):33–35, January 2005.

[50] A. Bruce McDonald and Taieb Znati. A Path Availability Model for Wireless Ad-Hoc Networks. *IEEE*, 1999.

[51] Tommaso Melodia and Francesca Cuomo. Ad Hoc Networking with Bluetooth: Key Metrics and Distributed Protocols for Scatternet Formation. *Ad Hoc Networks*, 2(2):185–202, 2004.

[52] G. Mühl. Generic Constraints for Content-based Publish/Subscribe. In *Cooperative Information Systems*, pages 211–225. Springer, 2001.

[53] G. Muhl, L. Fiege, F.C. Gartner, and A. Buchmann. Evaluating Advanced Routing Algorithms for Content-based Publish/Subscribe Systems. In *Modeling, Analysis and Simulation of Computer and Telecommunications Systems, 2002. MASCOTS 2002. Proceedings. 10th IEEE International Symposium on*, pages 167–176, 2002.

[54] Gero Muhl. *Large-Scale Content-Based Publish-Subscribe Systems*. PhD thesis, TU Darmstadt, 2002.

[55] Nokia. Introduction To Developing Networked MIDlets Using Bluetooth. print (2004), available at `http://fivedots.coe.psu.ac.th`.

[56] L. B. Oliveira, I. G. Siqueira, and A. A. F. Loureiro. On the Performance of Ad Hoc Touting Protocols under a Peer-to-Peer Application. *Journal of Parallel and Distributed Computing*, 65(11):1337–1347, 2005.

[57] M. Papadopouli and H. Schulzrinne. A Performance Analysis of Ë: A Peer-to-Peer Data Dissemination and Prefetching Tool for Mobile Users. `http://www.csd.uoc.gr/~hy544/mini_projects/Project13/sarnoff01.pdf`, 2001.

[58] C.E. Perkins and E.M. Royer. Ad-hoc On-demand Distance Vector Routing. In *Mobile Computing Systems and Applications, 1999. Proceedings. WMCSA '99. Second IEEE Workshop on*, pages 90–100, Feb 1999.

[59] C. Petrioli, S. Basagni, and M. Chlamtac. Configuring BlueStars: Multihop Scatternet Formation for Bluetooth Networks. *Computers, IEEE Transactions on*, 52(6):779–790, June 2003.

[60] Sundaram Rajagopalan and Chien-Chung Shen. A Cross-layer Decentralized BitTorrent for Mobile Ad hoc Networks. In *Mobile and Ubiquitous Systems: Networking Services, 2006 Third Annual International Conference on*, pages 1–10, July 2006.

[61] Sylvia Ratnasamy, Paul Francis, Mark Handley, and Richard Karp. A Scalable Content-Addressable Network. *ACM SIGCOMM Computer Communication Review - Proceedings of the 2001 SIGCOMM conference*, 31(4):40–48, 2001.

[62] M. Ripeanu, I. Foster, and A. Iamnitchi. Mapping the Gnutella Network: Properties of Large-Scale Peer-to-Peer Systems and Implications for System Design. *IEEE Internet Computing Journal*, 2002.

[63] Antony Rowstron and Peter Druschel. Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. In Rachid Guerraoui, editor, *Middleware 2001*, volume 2218 of *Lecture Notes in Computer Science*, pages 329–350. Springer Berlin Heidelberg, 2001.

[64] R. Schollmeier, I. Gruber, and M. Finkenzeller. Routing in Mobile Ad Hoc and Peer-to-Peer Networks - A Comparison. *Web Engineering and Peer to Peer Computing*, 2376:172–186, 2002.

[65] R. Schollmeier, I. Gruber, and F. Niethammer. Protocol for Peer-to-Peer Networking in Mobile Environments. In *Computer Communications and Networks, 2003. ICCCN 2003. Proceedings. The 12th International Conference on*, pages 121–127, Oct. 2003.

[66] S. Shahbazi, A. Harwood, and S. Karunasekera. An Analytical Model for Performance Evaluation in Sparse Mobile Ad Hoc Networks. In *Wireless Days (WD), 2009 2nd IFIP*, pages 1–6, 15–17 Dec 2009.

[67] Rui Shen, Ji Wang, Shengdong Zhang, Siqi Shen, and Pei Fan. A Framework for Constructing Peer-to-Peer Overlay Networks in Java. *PPPJ '09 Proceedings of the 7th International Conference on Principles and Practice of Programming in Java*, pages 40–48, 2009.

[68] D. Son, A. Helmy, and B. Krishnamachari. The Effect of Mobility-induced Location Errors on Geographic Routing in Mobile Ad Hoc Sensor Networks: Analysis and Improvement using Mobility Prediction. *Mobile Computing, IEEE Transactions on*, 3(3):233–245, July–Aug. 2004.

[69] S. Srinivasan, A. Moghadam, Se Gi Hong, and H. Schulzrinne. 7DS - Node Cooperation and Information Exchange in Mostly Disconnected Networks. In *Communications, 2007. ICC '07. IEEE International Conference on*, pages 3921 – 3927, June 2007.

[70] I. Stoica, R. Morris, D. Liben-Nowell, D.R. Karger, M.F. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications. *IEEE/ACM Transactions on Networking*, 11(1):17–32, Feb 2003.

[71] Ion Stoica, Robert Morris, David Liben-Nowell, David R. Karger, M. Frans Kasshoek, Frank Dabek, and Hari Balkrishnan. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. *Special Interest Group on Data Communication (SIGCOMM)*, 2001.

[72] M. Tarique, K. E. Tepe, S. Adibi, and S. Erfani. Survey of Multipath Routing Protocols for Mobile Ad Hoc Networks. *Journal of Network and Computer Applications*, 32:1125–1143, 2009.

[73] J. Tsumochi, K. Masayama, H. Uehara, and M. Yokoyama. Impact of Mobility Metric on Routing Protocols for Mobile Ad Hoc Networks. In *Communications, Computers and signal Processing, 2003. PACRIM. 2003 IEEE Pacific Rim Conference on*, volume 1, pages 322–325, Aug. 2003.

[74] Zhifang Wang, R.J. Thomas, and Z. Haas. Bluenet - A New Scatternet Formation Scheme. In *System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference on*, Jan. 2002.

[75] Brad Williams and Tracy Camp. Comparison of Broadcasting Techniques for Mobile Ad Hoc Networks. In *Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking & Computing*, MobiHoc '02, pages 194–205, New York, NY, USA, 2002. ACM.

[76] Bo Xu, A. Ouksel, and O. Wolfson. Opportunistic Resource Exchange in Inter-Vehicle Ad-Hoc Networks. In *Mobile Data Management, 2004. Proceedings. 2004 IEEE International Conference on*, pages 4–12, 2004.

[77] Sanlin Xu, Kim L. Blackmore, and Haley M. Jones. An Analysis Framework for Mobility Metrics in Mobile Ad Hoc Networks. *EURASIP Journal on Wireless Communications and Networking*, 2007(1):26–26, Jan 2007.

[78] Sanghyun Yoo, Jin Hyun Son, and Myoung Ho Kim. A Scalable Publish/Subscribe System for Large Mobile Ad hoc Networks. *Journal of Systems and Software*, 82(7):1152–1162, 2009.

[79] Dan Yu, Hui Li, and I. Gruber. Path availability in ad hoc network. In *Telecommunications, 2003. ICT 2003. 10th International Conference on*, volume 1, pages 383–387, Feb–Mar 2003.

[80] G.V. Zaruba, S. Basagni, and I. Chlamtac. Bluetrees-Scatternet Formation to Enable Bluetooth-Based Ad Hoc Networks. In *IEEE International Conference on Communications*, volume 1, pages 273–277, May 2001.

[81] Zhi Zhang and Ping Liu. Application of Bluetooth Technology in Ambulatory Wireless Medical Monitoring. In *Microwave and Millimeter Wave Technology, 2004. ICMMT 4th International Conference on, Proceedings*, pages 974–977, Aug. 2004.

[82] H. Zimmermann. OSI Reference Model–The ISO Model of Architecture for Open Systems Interconnection. *Communications, IEEE Transactions on*, 28(4):425–432, April 1980.

Appendix A

Data

Summarised Data

Total Traffic

Broadcasting	2	3	5	8
	2671	2671	2671	2671
	4106	8332	8332	8332
	2400	34066	34066	34066
	2816	88822	88822	88822
	2566	10148	9130	29732
	2489	13614	24539	9392
	4243	124	10014	35027
	2400	8472	15220	29334
	2403	9915	5324	31227
	4726	14686	9249	434
	7290	13723	26992	77377
	6726	7030	29236	45000
	5374	11239	44967	45862
	6562	26050	59162	93039
	18548	4810	18178	46322
	5790	26922	10007	60184
	6666	19968	57355	42589
	6874	13735	57355	121421
	5374	15886	49507	26553
	5894	23305	53907	54081
AODV	4664	4241	9427	14181
	738	1928	6593	16111
	1791	3445	2373	13495
	1002	3574	7694	5115
	1036	4670	9799	12083
	1127	5700	14832	14181
	2168	4148	5577	10824
	646	3150	2462	6101
	1162	3920	8713	6708
	718	1732	9388	3122
	11828	21622	42388	134430
	12050	56544	108580	190868
	10494	23212	153172	124562
	10962	45366	73004	48374
	1924	60894	42932	48468
	37068	22102	99908	189070
	11574	25968	13574	125532
	10270	58144	76960	263566
	72008	44383	42274	11886
	10062	26388	47194	142090

DSR	1797	2042	8119	27552
	1036	1860	5557	5042
	1054	1448	1448	6864
	1021	2883	7822	24083
	982	176	2317	616
	432	1580	5782	28182
	786	731	352	8368
	88	1466	8736	10128
	2541	1410	1444	12581
	348	2948	3937	2491
	7808	16736	66728	82356
	7808	19576	96616	122404
	3195	38784	35528	214144
	7860	29820	95944	190504
	8184	138800	95848	68228
	3704	16736	37140	117564
	7860	55864	89764	38820
	9184	16840	66728	38820
	8276	17360	34488	117676
	7808	17672	35528	117676
Content-based	2585	7929	12300	25518
	150	1804	13154	13165
	2578	3020	3292	24254
	1318	124	248	18574
	1422	2448	4962	7238
	1510	3876	7044	434
	62	464	600	6012
	2321	4510	4368	7972
	2691	1745	7044	9944
	2255	2417	5632	3066
	6726	25715	38989	46329
	5374	26340	23232	66838
	6562	13748	39196	26879
	6562	24476	40028	47572
	5374	8982	25704	37874
	2402	17217	44041	78858
	3673	11824	22787	68294
	5894	14739	22787	50874
	4638	26340	11018	78858
	7169	20768	21724	30438

Data Traffic

Broadcasting	2609	2547	2423	2237
	4044	8208	8084	7898
	2338	33942	33818	33632
	2754	88698	88574	88388
	2504	10024	8882	29298
	2427	13490	24291	8958
	4181	0	9766	34593
	2338	8348	14972	28900
	2341	9791	5076	30793
	4664	14562	9001	0
	7228	13599	26744	76943
	6664	6906	28988	44566
	5312	11115	44719	45428
	6500	25926	58914	92605
	18486	4686	17930	45888
	5728	26798	9759	59750
	6604	19844	57107	42155
	6812	13611	57107	120987
	5312	15762	49259	26119
	5832	23181	53659	53647
AODV	1702	2911	6487	11171
	454	1104	4979	12739
	1433	2675	1259	9849
	772	2676	4892	3311
	860	4042	7537	8923
	931	4748	720	3699
	1898	3088	4395	7272
	430	2596	1172	3311
	912	2826	6883	4560
	522	1232	7710	1090
	10228	17288	34760	120984
	10396	53526	101640	174826
	8192	17184	135952	110844
	9308	42254	63256	36060
	0	57120	36424	35696
	35360	16560	93468	159738
	9704	20784	8032	111720
	8508	53304	66186	247668
	68464	40103	35800	24
	8408	22648	39700	115052

DSR	1709	1866	7767	26848
	948	1772	5205	4426
	966	1272	1096	6336
	933	2707	7470	23379
	894	0	1877	0
	344	1404	5406	27566
	698	507	0	7752
	0	1290	8384	9512
	2453	1234	1092	0
	260	2772	3673	0
	7720	16560	66376	81612
	7720	19400	96264	121740
	3107	38608	35176	213480
	7772	29688	95592	189928
	8096	138624	95496	67564
	3640	16560	36788	116968
	7772	55688	89412	38244
	9096	16664	66376	38244
	8188	17184	34136	117012
	7720	17496	34176	117012
Content-based	2523	7805	12548	25084
	88	1680	13402	12731
	2516	2896	3540	23820
	1256	0	496	18140
	1360	2324	5210	6804
	1448	3752	7292	0
	0	340	848	5578
	2259	4386	4616	7538
	2629	1621	7292	9510
	2193	2293	5880	2632
	6664	25591	39237	45895
	5312	26216	23480	66404
	6500	13624	39444	26445
	6500	24352	40276	47138
	5312	8858	25952	37440
	2340	17093	44289	78424
	3611	11700	23035	67860
	5832	14615	23035	50440
	4576	26216	11266	78424
	7107	20644	21972	30004

Delay

N = 2	Broadcasting	AODV	DSR	Content-based
	19	29	16	22
	15	52	27	15
	0	22	21	15
	25	18	26	0
	19	22	24	20
	14	19	31	20
	13	22	15	0
	0	20	0	15
	21	23	25	15
	13	31	46	0
	8	17	38	20
	4	17	38	6
	6	17	25	9
	9	32	47	9
	9	0	39	7
	4	32	0	0
	11	49	45	0
	7	21	18	4
	7	21	44	14
	11	36	25	9

N = 3	Broadcasting	AODV	DSR	Content-based
	19	26	21	0
	15	32	52	32
	11	29	21	0
	9	37	30	5
	8	26	0	19
	8	37	36	0
	0	27	28	15
	27	48	29	24
	17	35	27	34
	11	24	24	33
	2	38	18	30
	10	0	31	34
	6	37	35	19
	9	31	0	24
	0	20	38	16
	3	22	38	30
	4	15	25	34
	8	19	39	19
	8	24	45	24
	7	32	67	16

N = 5	Broadcasting	AODV	DSR	Content-based
	19	27	25	10
	15	38	29	9
	11	30	30	21
	9	25	26	0
	0	35	19	12
	16	34	36	19
	14	30	0	20
	11	19	26	3
	12	25	24	40
	12	29	72	5
	6	30	22	29
	11	36	31	15
	7	37	37	21
	7	39	86	32
	12	31	35	24
	9	41	21	17
	6	28	25	35
	7	28	30	35
	7	25	44	23
	4	26	18	7

N = 8	Broadcasting	AODV	DSR	Content-based
	6	27	35	10
	10	29	36	21
	17	32	17	12
	23	45	22	1
	8	43	0	12
	2	40	31	0
	2	38	83	0
	10	34	23	16
	11	30	23	16
	0	26	23	27
	5	32	57	27
	3	21	42	10
	4	152	23	25
	9	106	28	14
	6	105	26	40
	3	97	47	11
	11	93	27	18
	9	84	27	9
	3	0	35	16
	3	148	35	11

Positive Response

	2	3	5	8
Broadcasting	7.69231	0	0	0
	0	5.12821	5.2182	5.2182
	83.87097	1.36986	1.36986	1.36986
	96.8085	1.020408	1.020408	1.020408
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	50.58039	67.53
	0	0	55.319	47.13
	0	70.27	81.25	0
	0	0	74.2857	64.01
	0	71.2329	0	0
	0	0	0	59.77
	0	0	50.98	0
	0	0	83.87	46.846
	0	67.5325	67.53	0
	0	0	50.98	37.59
AODV	73	81	87	90
	75	67	88	96
	64	88	94	90
	57	80	88	96
	75	86	90	90
	63	83	80	85
	81	78	86	89
	50	84	80	87
	67	86	92	89
	60	75	91	75
	98	99	100	100
	98	100	100	100
	98	99	99	100
	98	100	100	100
	0	100	100	100
	100	99	100	100
	98	99	98	100
	98	100	100	100
	100	100	100	0
	98	99	100	99
DSR	73	75	90	93
	75	69	87	90

	70	75	80	96
	75	84	93	98
	75	0	56	100
	67	86	91	96
	83	75	0	94
	0	86	94	95
	73	75	80	95
	50	84	89	96
	98	99	100	99
	98	99	100	100
	90	100	100	100
	98	100	100	100
	98	100	100	100
	100	99	100	100
	98	100	100	100
	98	99	100	100
	98	99	100	100
	98	99	100	100
Content-based	0	6.25	12.5	8.612
	0	14.2857	3.305	11.22
	0	7.4074	13.79	2.59
	0	0	0	0
	0	0	4.347	9.09
	0	0	6.25	0
	0	0	0	0
	0	0	0	11.11
	0	16.67	6.25	13.846
	0	4.3478	7.4	0
	0	73.049	88.84	67.096
	0	74.82	67.54	93.57
	0	0	89.27	0
	0	63.8	89.27	100
	0	0	0	0
	0	44.23	90.43	96.808
	0	0	0	93.57
	0	0	0	83.87
	0	74.87	0	96.808
	4.7619	80.62	0	0

Control Traffic

Broadcasting	62	124	248	434
AODV	2962	1330	2940	3010
	284	824	1614	3372
	358	770	1114	3646
	230	898	2802	1804

	176	628	2262	3160
	196	952	14112	10482
	270	1060	1182	3552
	216	554	1290	2790
	250	1094	1830	2148
	196	500	1678	2032
	1600	4334	7628	13446
	1654	3018	6940	16042
	2302	6028	17220	13718
	1654	3112	9748	12314
	1924	3774	6508	12772
	1708	5542	6440	29332
	1870	5184	5542	13812
	1762	4840	10774	15898
	3544	4280	6474	11862
	1654	3740	7494	27038
DSR	88	176	352	704
	88	88	352	616
	88	176	352	528
	88	176	352	704
	88	176	440	616
	88	176	376	616
	88	224	352	616
	88	176	352	616
	88	176	352	12581
	88	176	264	2491
	88	176	352	744
	88	176	352	664
	88	176	352	664
	88	132	352	576
	88	176	352	664
	64	176	352	596
	88	176	352	576
	88	176	352	576
	88	176	352	664
	88	176	1352	664
Content-based	62	124	248	434